Projecting named entity recognizers from resource-rich to resource-poor languages without annotated or parallel corpora

Hou, Jue

Helsinki October 20, 2019

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Studieprogram — Study Programme
Faculty of Science	Computer Science

Tekijä — Författare — Author
Hou, Jue

Työn nimi — Arbetets titel — Title
Projecting named entity recognizers from resource-rich to resource-poor languages without annotated or parallel corpora

Ohjaajat — Handledare — Supervisors
Roman Yangarber

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	October 20, 2019	56 pages

Tiivistelmä — Referat — Abstract

Named entity recognition is a challenging task in the field of NLP. As other machine learning problems, it requires a large amount of data for training a workable model. It is still a problem for languages such as Finnish due to the lack of data in linguistic resources. In this thesis, I propose an approach to automatic annotation in Finnish with limited linguistic rules and data of resource-rich language, English, as reference. Training with BiLSTM-CRF model, the preliminary result shows that automatic annotation can produce annotated instances with high accuracy and the model can achieve good performance for Finnish.

In addition to automatic annotation and NER model training, to show the actual application of my Finnish NER model, two related experiments are conducted and discussed at the end of my thesis.

ACM Computing Classification System (CCS):
Computing methodologies → Machine learning → Machine learning approaches → Neural networks
Computing methodologies → Artificial intelligence → Natural language processing → Lexical semantics
Computing methodologies → Artificial intelligence → Natural language processing → Information extraction

Avainsanat — Nyckelord — Keywords
Neural Network, Natural Language Processing, Named Entity Recognition

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Thesis for the Algorithms, Data Analytics and Machine Learning subprogramme

Contents

1 Introduction

As one important part of natural language processing (NLP), information extraction (IE) is the task of extracting key information from unstructured human language data (text) and make it available for further machine processing or better data representation. Systems for different purposes can be built on these. My work is closely related to PULS project [16], an IE system focused on the business domain.[1] One of my current tasks is to resolve the locality of Finnish news. So in my master's thesis, I will focus on one of IE sub-tasks: Named Entity Recognition (NER).

The goal of NER is not only to identify every name from texts, but also to tell the type of that name, such as persons, organizations or locations. When we—as human readers—read texts, we utilize text context, previous knowledge, and common sense. Taking advantage of such information, we can identify names and then deduce their type. However, for machines, things are more complicated. We can not just define some naive rules to map a name to a certain type. Nor can we collect all names in this world and store them in a database as reference. And sometimes, even the same name may refer to different meanings according to its context. For example, Coco Chanel can both mean a famous fashion designer or a luxury brand and its products.

Researchers have already made several successful attempts to solve the task of NER in different languages [16, 18, 20, 29, 31, 24]. Currently, the state of the art for NER is a model based on attention mechanism [42, 4]. However, all of these researches are conducted for major languages such as English, German and Spanish. Finnish, as a minor language, is unfortunately not a part of their primary goal.

On the other hand, as many approaches as people have explored, there seems to lack resources for Finnish as well. FiNER-data [39] is one of few publicly available datasets for Finnish NER model training that I can find online at the moment.[2] And it is in a relatively small size with only technology-related news covered.

Therefore, in this thesis, there are two major issues that I try to address. One is to propose a pipeline of automatic annotation on Finnish named entities so that a Finnish NER dataset can be generated. By utilizing an existing English NER tagger, English named entities and their corresponding types are used as the source of annotations. The idea is to project the *types* of the named entities from English to

[1] `http://puls.cs.helsinki.fi`

[2] `https://github.com/mpsilfve/finer-data`

Finnish. The projection is done by resolving and matching the base forms of named entities. Another task is to implement an NER model and evaluate its performance given the data generated by the previous pipeline. This can be viewed as a projection between an existing English NER tagger to a new Finnish NER tagger (Proj-NER). To show the application of my Proj-NER, two related experiments are also conducted and discussed at the end of my thesis. This thesis is organized as follows:

- Section 2 will give an overview of automatic named entity annotation. Several classic NER approaches and the state-of-the-art of NER will be discussed as well.
- Section 3 will introduce the background knowledge of my thesis, including four critical linguistic terms and the metrics that is used in this thesis. I will also clarify the goal of automatic named entity annotation, Finnish NER, and two additional experiments and give an overview of the methodology in this section.
- Section 4 will give a discussion on the whole pipeline for data generation, starting from a raw text to the output of annotated named entities. The quality of English NER taggers will also be discussed here.
- Section 5 will provide details for training a Finnish NER model. The hardware environment and setup parameter of the model will be introduced.
- Section 6 will discuss the performance of both the automatic name annotation pipeline and my Proj-NER models respectively.
- Section 7 will introduce two additional experiments of my Proj-NER tagger. One is about locality visualization. The name linking step and setup for locality visualization will be discussed in this section. Examples of visualization will also be demonstrated here. The other experiment will show the influence of named entities and different name representation on the quality of document representation.
- Section 8 will summarize the contribution of my thesis and discuss future work briefly.

2 Related Work

There are two major parts in my work: automatic data annotation and a Finnish NER model. For automatic annotation, several approaches have been proposed. One common approach is to extract named entities and their corresponding annotations from Wikipedia [2, 19, 25, 26, 38, 34, 41]. By utilizing the meta-data of Wikipedia documents or the links between documents and linked elements, named entities can be identified. Most of the research relies on language-specific techniques and parallel corpora. As a consequence, they can produce NER datasets for only several languages. Finnish NER corpora are not one of them. Ehrmann et al. [17] proposed an idea of model projection similar to the one in this work. Rather than resolving the base form of named entities in the target language internally, as done in this thesis, they used machine translation as the basis for projection. This allows them to project models between different languages, including between languages with different writing systems, such as Russian and English. However, this also means that the availability and the quality of the machine translation component are critical for the quality of the resulting training dataset.

Compared to the limited resources and approaches in the field of automatic data annotation, the field of NER is more well-researched. Researchers have applied different approaches to this task. They can be categorized as pattern-based approaches, statistical model-based approaches, neural network-based approaches and hybrid approaches. Most of these implementations are closely related to my project, and I will give a detailed introduction in Section 3. In this section, I will give only an overview of each approach and the state-of-the-art implementation.

One classic statistical NER approach is the Stanford NER CRF model [18]. CRF is a statistical probabilistic model. This approach is effective in English, German, Spanish and Chinese.

With the rise of deep learning, researchers proposed NER models on the basis of neural networks. Collobert et al. [11] and Al-Rfou et al. [2] tackle the NER task as token-level classification. They utilize a simple feed-forward neural network model, which classifies tokens independently by using the information of the neighbour of each token in a fixed window.

Compared to simple feed-forward models, RNN-based neural network models, such as the model proposed by Chiu and Nichols [10], has proved to be more effective. There are approaches combining the CRF model and neural network based mod-

els [31, 29, 37]. These models are fed with contextualized embeddings, which is a concatenation of several features, including word embeddings, character embeddings, case information and so on. Researchers have additionally explored different word embedding setups as an enhancement to contextualized embeddings [32, 35, 36, 1]. All of them achieve increasingly better performance, but they also require more computational resources.

Recently, large-scale neural networks with attention mechanism [5] achieved state-of-the-art performance. The idea of attention is to let the model select key features by plugging in additional feature weights. Based on this mechanism, a special encoder-decoder network with self-attention unit—the "transformer"—has been proposed [42]. Devlin et al. [14] implement a model called BERT on the basis of bi-directional transformers. Baevski et al. [4] used multi-head transformers to achieve the state-of-the-art scores in the NER task. The computational resources for training, however, are even more expensive. Baevski et al. [4] allocated 128 Volta GPUs and spent on average a week for training models.

In addition to all the above approaches, pattern mining can also be used to solve NER, by assigning a role to each token according to a set of pre-defined patterns. Such a role can indicate the type of the name. One example implementation is from the PULS media monitoring system [16].

In this thesis, taking into account our limited computational resources, which are not capable of training large-scale neural networks, my Proj-NER tagger is implemented on the basis of BiLSTM-CRF. I use context-independent word embeddings, such as Word2Vec, since they can be obtained within shorter time and allocate less memory, compared to contextual embeddings, such as ELMo or Flair. The main results of this thesis have been published and presented at NoDaLiDa-2019—The 22nd Nordic Conference on Computational Linguistics [22].

3 Background

In this section, I discuss all necessary topics and concepts that are related to my thesis. The purpose of doing so is to give a background to NER and my work.

3.1 Terminology

Token A token is an abstraction which is defined by external tools or rules and is used as an atomic unit of processing in NLP tasks. Typically, a token is defined as a string of characters between two spaces or between a space and a punctuation mark. A token can be a word, a number, an acronym or punctuation. In this thesis, tokens mostly refer to the words which appear in a sentence.

Base form The base form of a word, also referred to as **lemma**, is the canonical, or "dictionary," form of a word. As an example, the base form of the English verb "was" is "be".

Surface form The surface form of a token is the form in which the word appears in the actual text. The surface form may be inflected, such as "was", or be identical with its base form, such as "run".

Compound A compound is a word which consists of multiple "free morphemes" or "roots" which can stand on their own.[3] For example, "pancake" consists of two parts: "pan" and "cake". Some languages, such as Finnish, make extensive use of compound words.

Part-of-Speech The Part-of-Speech (PoS) of a word is its morpho-syntactic category or class. This indicates the role that a word plays in a sentence as well as the pattern of inflection that the word follows. Examples of PoS are: noun, verb, and adjective.

3.2 Metrics

Salience Du et al. [16] proposed *salience* as a metric for names. It can evaluate and quantify how salient a name is with respect to an article. *Salience* is defined as follows:

[3] As opposed to "bound" morphemes, which cannot stand on their own, and must appear together with other morphemes to form a word. For example: in English "cleaner", "clean" is a free and "er" is a bound morpheme; in Finnish "talossa"—"talo" is free, and "ssa" is bound.

$$salience_w = \sqrt{\frac{|S| - fp_w}{|S|} \cdot \frac{n_w}{|D|}} \tag{1}$$

$|S|$ = the number of sentences within current article. $|S| > 0$

fp_w = the index of sentence where word w first appears in the current article. $0 \leq fp_w \leq |S| - 1$

$|D|$ = the number of extracted words that the current article has. $|D| > 0$

n_w = the count of words w within the current article. $0 \leq n_w \leq |D|$

Therefore, $0 < \frac{|S|-fp_w}{|S|} \leq 1$, $0 \leq \frac{n_w}{|D|} \leq 1$ and $0 \leq salience_w \leq 1$.

Salience is based on the assumption that frequent names or the names that appear at the beginning of an article are more salient than the others. This assumption is based on common practices of modern journalism. Journalists put the most important names or their point of view at the beginning of an article. Also, a name that is mentioned many times in the article is more important than the names that only appear one or two times.

TF-IDF Term Frequency-Inverse Document Frequency (TF-IDF) is one common metric to evaluate the importance of a word with respect to a document in a corpus. It is a product of two factors: Term Frequency (TF) and Inverse Document Frequency (IDF). In practice, there are several definitions for both TF and IDF that can be applied in different circumstances. In this thesis, the most classic definition is used, which can be formalized as follows:

$$tf(w, D) = \frac{n_w}{|D|} \tag{2}$$

$$idf(w, C) = \log \frac{|C|}{|D : w \in D|} \tag{3}$$

n_w is the number of occurrences of word w in document d.

$|D|$ is the total number of words in document d.

$|C|$ is the size of corpus C.

$|D : w \in D|$ is the number of documents that contain the word w.

Though TF-IDF can measure the overall significance of a word, it fails to measure the importance of the word according to the assumption mentioned previously, which

is that important words appear first. Inspired by *salience*, I introduce the first prominence factor into the formula and refine TF-IDF as follows:

$$sal_{tfidf}(w, D, C) = \sqrt{\frac{|S| - fp_w}{|S|} \cdot tf(w, D) \cdot idf(w, C)} \tag{4}$$

F1-score measures the quality of prediction. It is a combination of *precision* and *recall*. They can be defined as follows:

$$precision = \frac{TP}{TP + FP} \tag{5}$$

$$recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{7}$$

Assume there are many entities and the task is to classify whether they belong to class c.

TP = true positives: the number of correctly classified entities that are in class c.

FP = false positives: the number of class c entities which are misclassified to not belonging to class c.

FN = false negatives: the number of entities which are misclassified to class c.

The value of *precision*, *recall* and $f1$-*score* ranges from zero to one. This is based on a binary classification problem. In multi-class classification, the formulas can be applied in a similar way. In my thesis, **weighted average f1-score** is used as an overall evaluation of performance.

3.3 Problem Formulation

3.3.1 Automatic Named Entity Annotation

The goal of automatic named entity annotation is to create a large amount of NER instances in a corpus for training models and to filter out any uncertain instances.

To achieve this goal, the base forms of Finnish named entities are resolved, matched and projected with the type of English named entities, which are tagged by an

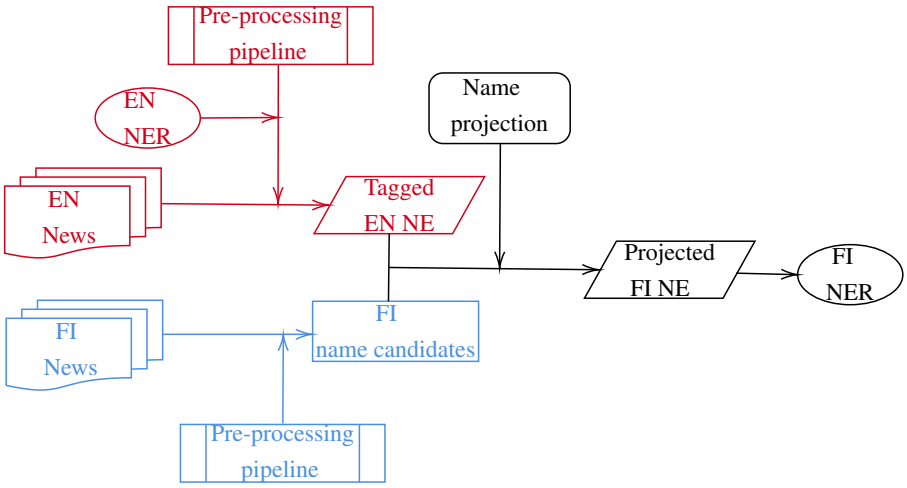

Figure 1: The pipeline of Proj-NER

existing English NER model. Based on modern journalism practice, a series of assumptions and rules are applied to resolve the base form of names, link names and filter out dirty data. Taking advantage of our enormous amount of articles in both English and Finnish, any uncertain data can be filtered out without worrying about the lack of data. Figure 1 is a diagram of my Proj-NER pipeline.

Tagging Scheme There are several types of tagging scheme that both datasets and models have to follow. Table 1 shows examples of different tagging schemes. **IO** scheme is the simplest and most straightforward scheme. It tags tokens either as "I" only if they appear inside of a name or "O" only if they are not. However, when it comes to further processing, the lack of border between two different names will make it difficult to separate. **IBO** and **IBO2** tag the beginning name token as "B". The only difference between these two schemes is that IBO2 tags any beginning name token as "B", while IBO tags a token "B" only if this token is followed by a token from the same name. A name token which is surrounded by "O" tokens will still be tagged "I". As illustrated in Table 1, "Mary" and "Washington" are tagged differently according to IBO and IBO2. **BMEWO** further distinguish entity borders besides beginning border "B" tag. It will tag the middle tokens of an entity as "M" and the ending token of an entity as "E". For entities of a single token, they will be tagged as W. In my thesis, IBO2 tag is used for its simplicity.

Tokens	*IO*	*IBO*	*IBO2*	*BMEWO*
Yesterday	O	O	O	O
afternoon	O	O	O	O
,	O	O	O	O
Mary	I-PER	I-PER	B-PER	W-PER
and	O	O	O	O
her	O	O	O	O
husband	O	O	O	O
John	I-PER	B-PER	B-PER	B-PER
J	I-PER	I-PER	I-PER	M-PER
.	I-PER	I-PER	I-PER	M-PER
Smith	I-PER	I-PER	I-PER	E-PER
traveled	O	O	O	O
to	O	O	O	O
Washington	I-LOC	I-LOC	B-LOC	W-LOC
.	O	O	O	O

Table 1: NER tagging scheme

3.3.2 NER Model

In this part, the goal is to implement and evaluate a Finnish NER model as a part of my Proj-NER. The NER model and its setup are inspired by the models introduced and evaluated by Ma and Hovy [31] and Reimers and Gurevych [37]. My model is modified from the original BiLSTM-CNN-CRF model, which is proposed by Ma and Hovy [31]. Compared to the feature mentioned in the original articles, PoS information is utilized as one additional input features. Hopefully, it can improve the performance of the model.

Manual checking is conducted to evaluate the true performance of the NER model after training. Besides regular model performance evaluation, the quality of different English NER taggers, such as rule-based NER tagger or neural network based NER tagger, and their influence on the performance of Proj-NER is a part of the evaluation as well.

3.3.3 Locality Resolution

In this part, the goal is to resolve all the location names and assign each article the most salient location.

After resolving all the named entities of an article, the metric *salience* is used to select the most important location name as the locality of an article. Articles are plotted on a map as visualization according to their locality. A location-coordinate dictionary is used for obtaining the coordinate of a geo-location.

3.3.4 Named Entities in Document Representation

The goal of this experiment is to answer one question: Does the representation of named entities contribute to the quality of document representation?

The same indirect evaluation method used by Chen [9] is adopted, which is to solve a document classification task. By comparing the performance of document representations with different named entity setups, researchers can get a better understanding of the role of named entities in document representation and how to handle them to achieve better performance.

3.4 Neural Network

I will give a brief introduction to three typical neural networks: MLP, CNN, and RNN.

The idea of neural networks is inspired by the way how the brain works. It is a framework instead of a single algorithm as other machine learning approaches. One example is **multi-layer perceptron** (MLP). It is one of the most fundamental types of neural network. Such networks have an input layer, an output layer, and several hidden layers. Each layer has several nodes (neurons). Each neuron takes a weighted sum of the outputs from its previous layer and passes it to its next layer after applying an activation function. Examples of activation functions are sigmoid ($\sigma(x)$) and $tanh(x)$.

Figure 2 is a simple example of MLP with only one hidden layer. A single hidden layer network can be formalized as $\hat{y} = g(Vh(Wx))$. W and V are two weight matrices, $h(\cdot)$ and $g(\cdot)$ are two activation functions.

To train an MLP, back-propagation will be applied to adjust the weight matrix

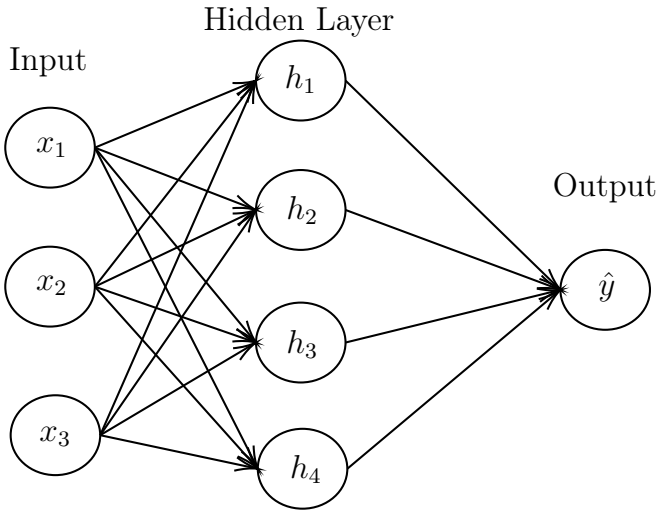

Figure 2: An example of MLP network

for each layer. The objective function in the training—the loss—is the difference between y and $\hat{y}$, where y is the ground truth and $\hat{y}$ is the output of the MLP. Essentially, back-propagation computes gradients on each neuron with respect to later neurons using the chain rule. Because of the chain rule, the loss is distributed across the whole network. Back-propagation adjusts the weights of all neurons in a network so that the loss can be reduced.

Convolutional Neural Network (CNN) utilizes convolution as a part of a network and obtains the information of the local neighbourhood of each neuron. CNN is therefore capable of solving tasks like image recognition.

Convolution on 1-dimensional space can be discretely formalized as $(f * g)(t) = \sum_m f(t-m)g(m)$. Function $f(\cdot)$ refers to an input, while function $g(\cdot)$ refers to a kernel function. The output of a convolution is referred as a feature map. Convolution can be extended to 2-dimensional space or more.

Besides convolution layer, CNN introduces two pooling layers: max pooling and average pooling. They scan over the feature map produced by the previous layer and extract max or average values of each neighbourhood.

Figure 3 is one example of CNN. A 2D Conv filter of size 2×2 and stride 1 scans over a 2D matrix and outputs a 1D feature map. A 1D max pooling layer of stride 1 extracts the maximum value of each neighbourhood.

Although the output of the example shown in Figure 3 is of 1D and ready to be sent into 1D feed-forward layers, there will usually be an additional layer between

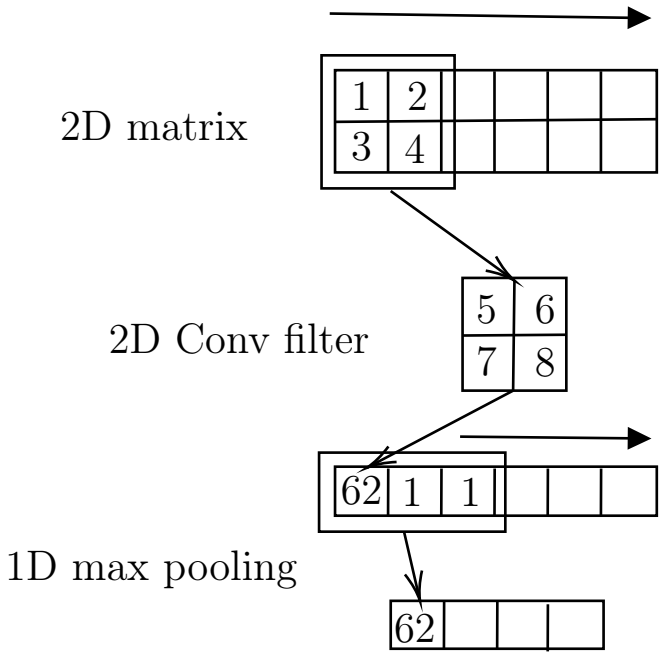

Figure 3: An example of CNN Application

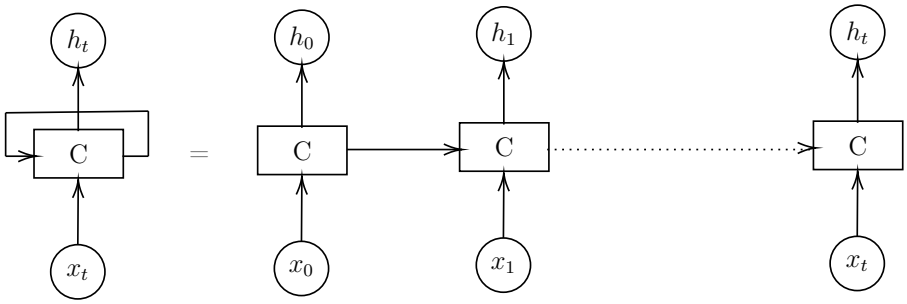

Figure 4: An example of RNN

a Conv layer and a 1D feed-forward layer to flatten multi-dimensional feature map into a 1D array.

MLP and CNN are considered as feed-forward networks. **Recurrent Neural Network** (RNN) utilizes recurrent links, which allows the output of each neuron to be linked back from the previous time step. Because of this feature, RNN can be applied for tasks such as sequence tagging.

Figure 4 is a simple illustration of RNN. The left side of Figure 4 shows how recurrent link connects RNN cell back to itself after computing the "cell state" (hidden state). The right side illustrates how the recurrent connection passes information over time. Here, x_t refers to the input, h_t refers to the output with respect to the corresponding input x_t, and C refers to the cell state.

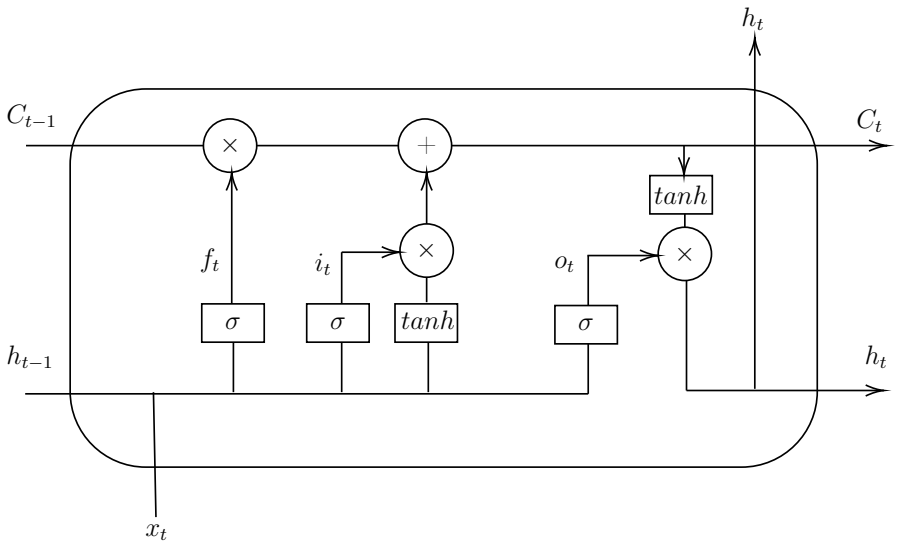

Figure 5: An example of LSTM unit

There is a problem which comes along with fitting through a sequence: vanishing gradient. The gradients may be close to zero or explode after fitting with a long sequence. To solve this problem, researchers have proposed several RNN cell structures. One of them is Long Short-Term Memory (LSTM).

As shown in Figure 5, an LSTM unit is an RNN cell which is composed of a forget gate, an input gate, and an output gate. They can be formalized as follows:

- Forget gate: $f_t = \sigma(U_f x_t + W_f h_{t-1})$
- Input gate: $i_t = \sigma(U_i x_t + W_i h_{t-1})$
- Output gate: $o_t = \sigma(U_o x_t + W_o h_{t-1})$
- Matrices U and W are weights of x_t and h_{t-1} for different gates

In Figure 5, C_t and C_{t-1} refer to the cell state, where C_{t-1} can be updated to C_t by applying the input and forget gates: $C_t = f_t \times C_{t-1} + i_t \times tanh(U_c x_t + W_c h_{t-1})$, where W_c and U_c are another pair of weight matrices. The symbol $\times$ refers to element-wise vector multiplication. The output of the cell will be filtered by output gate: $h_t = o_t \times tanh(C_t)$. The idea of "forgetting" is to decide what information should be thrown away. With the help of "forgetting", vanishing gradient problem is solved.[4]

[4]Figure 4, Figure 5 and gate formulas are inspired by the online blog post `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

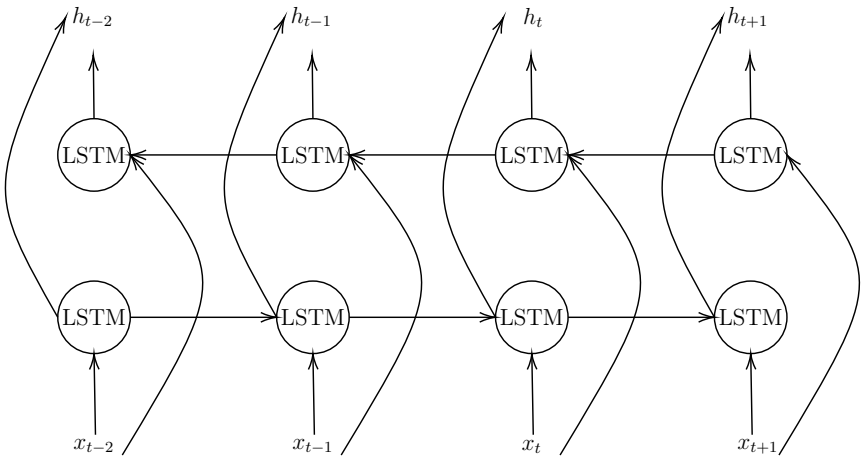

Figure 6: The structure of BiLSTM

Another problem in RNN is that a regular RNN network runs only in one direction, which is usually chronological direction. This works well for single-directional data, such as heart rate sequence. But what about the bi-directional data? For a text sequence, information of different parts of the sequence can be inter-connected from both forward and backward directions.

To solve this problem, Bi-direction LSTM (BiLSTM) is proposed on the basis of LSTM. The idea of BiLSTM is to utilize two LSTM layers for forward and backward directions of a sequence and concatenate the outputs of two LSTM layers. Figure 6 is a diagram of BiLSTM.

3.5 Word Embedding

Word embedding is one type of representation of document vocabulary. It has now been implemented in many NLP applications. Compared to other statistical language models (SLM), such as N-gram, word embedding will not learn the probability distribution of words in different contexts but the vector representation of each word. Several word embedding approaches will be briefly discussed in this subsection.

One way of obtaining an embedding is to use a neural network. Google's Word2Vec [32] is one well-known example. There are two architectures for training a Word2Vec embedding. As shown in Figure 7, both of them contain one input layer, one hidden layer, and one output layer, and the training process is unsupervised. The CBOW model will take the surrounding context of word $w(t)$ as input and try to predict $w(t)$, while the Skip-gram model will output the surrounding words, given word

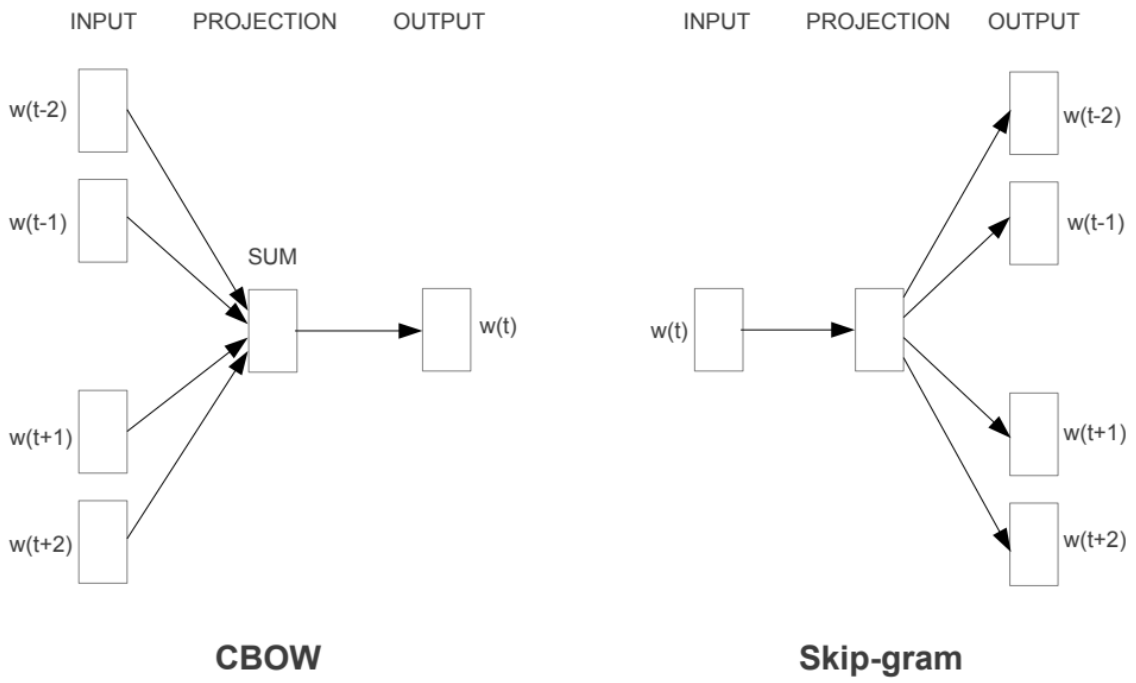

Figure 7: Neural Networks for Word2Vec [32]

$w(t)$.

Word2Vec covers only word-level embeddings, which leads to its poor performance for morphologically-rich languages. Such languages make extensive use of prefixes, suffixes and compound words. One base form may be inflected to many surface forms. Therefore, it is impossible to have enough data to train a good embedding for the surface forms.

To solve this shortcoming, Facebook's Fasttext [8] is proposed. It is built on an idea similar to Word2Vec, but it uses all substrings of a word. For example, the word "disagree" and its substrings "dis", "isa", "sag", "agr", "gre", and "ree" are all included in Fasttext training process. This approach leads to better support for morphologically-rich languages and solves the out-of-vocabulary problem. However, this also suggests that Fasttext needs more memory allocation and computational power in training and actual usage, compared to Word2Vec.

GloVe [35] takes a different approach to word embedding, which is statistical. It first builds a word-word co-occurrence matrix and defined an approximate equation between the co-occurrence probability of words w_i and w_j and their word vectors. Then, it addresses the problem as a weighted least squares regression model with

other conditions. By optimizing a weighted loss function, the word representation is obtained. Similarly to Word2Vec, GloVe takes advantage of the local context window. However, GloVe also takes global corpus information into consideration through the word-word co-occurrence matrix, while Word2Vec and Fasttext do not [35].

Researchers have also proposed more advanced word embeddings. ELMo [36] and Flair [1] are two examples. Both are implemented on the basis of BiLSTM. ELMo is still a word-level embedding, while Flair is based on the concatenation of character-level embedding. Compared to Word2Vec, Fasttext and GloVe, ELMo and Flair produce context-specific word embeddings. A token may be assigned with different embedding according to its context. This is an advantage for disambiguation. But on the other hand, utilizing BiLSTM leads to a more time-consuming and memory-intensive training process.

One of the goals and benefits of word representation is to measure the similarity of words. For example, "hound" and "dog" should be close to each other. Another benefit of applying embeddings is that it can help to obtain analogy. In Word2Vec, the most similar word to $vec(\text{"}king\text{"}) - vec(\text{"}man\text{"}) + vec(\text{"}woman\text{"})$ is $vec(\text{"}queen\text{"})$.

For the task of NER, the quality of embedding is essential. The better the word embedding, the better the result will be [37, 36, 1]. However, due to the fact that ELMo and Flair are very time-consuming and memory-intensive, only Word2Vec and GloVe will be used in my thesis.

3.6 Document Representation

Similarly to word representation, document representation, also referred as document embedding, maps a document to an n-dimensional vector. It is concerned about how text should be represented in different problems, such as topic classification and measuring the similarity of documents.

One approach to build document representation is to use a weighted average of the word embeddings. The weight of a word embedding can either be the frequency of its corresponding word in an article, or the inverse document frequency (IDF) with respect to a corpus. Each dimension of this document embedding can be regarded as a summary of the same dimension of the word embeddings. As a whole, a document embedding, therefore, summarizes its corresponding document.

Another approach is to use deep learning. Doc2Vec [30] is one example, which

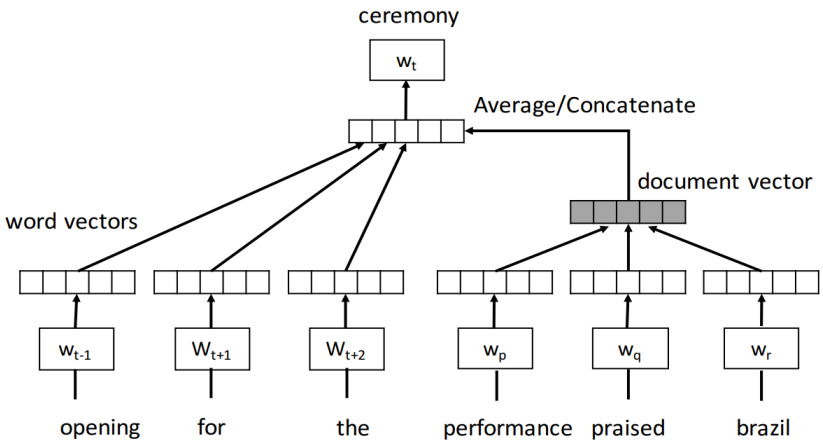

Figure 8: The framework of learning Doc2VecC [9]

introduces two models: Distributed Memory Model of Paragraph Vectors (PV-DM) and Distributed Bag of Words version of Paragraph Vectors (PV-DBOW). PV-DM takes the surrounding context tokens and an additional paragraph vector as an input and predicts the center word, while PV-DBOW takes only paragraph vector and predicts context tokens. Both models are initialized randomly and train paragraph vectors directly. Others have also proposed indirect approaches, where a paragraph embedding is assembled with certain rules. Huang et al. [23] improve the PV-DM model by utilizing globally averaged word embeddings as the paragraph vector. Although the goal is to improve the quality of word embeddings by introducing both local context and global context into training, the paragraph embedding will benefit from this approach as well. Similarly, in Doc2VecC [9], the paragraph embedding is assembled via averaging word embeddings of randomly sampled words from a document. Despite the "corrupted" document representation due to dropping a significant portion of words, this model outperforms other approaches in several tests. Figure 8 is an illustration of training a Doc2VecC model.

3.7 Conditional Random Field

Conditional Random Field (CRF) is a discriminative probabilistic model. One statistical approach for NER, Stanford NER, is built on CRF and has been proved to be effective in many different languages [18].

Heavy mathematical detail will be omitted in this section. Only fundamental con-

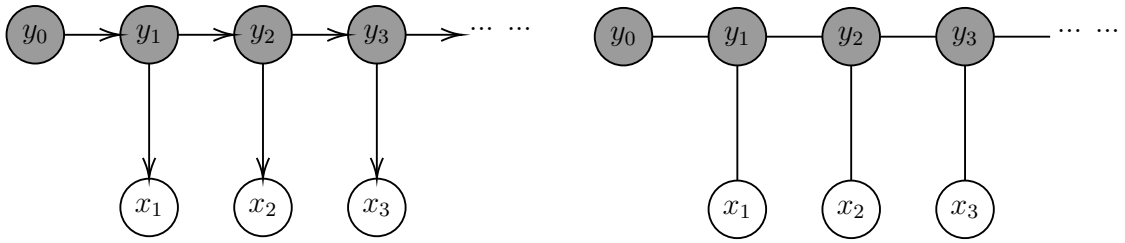

Figure 9: HMM and Linear CRF [40]

cept and application will be introduced here, especially in the NER scenario.[5]

Structurally, linear CRF model is very similar to Hidden Markov Model (HMM), but HMM is a Directed Acyclic Graph (DAG), while CRF is undirected. Figure 9 illustrates the structure of HMM and linear CRF. For both models, the shaded nodes are the hidden state variables and the light nodes represent the observed variables. For HMM, an observed variable depends on its corresponding hidden variable, while a hidden variable depends on its previous hidden variable. Using the chain rule, the joint probability of HMM can be decomposed as follows:

$$P(X,Y) = P(y_0)\prod_{t=1}^{T-1} P(y_{t+1}|y_t)\prod_{t=1}^{T} P(x_t|y_t) \tag{8}$$

In (8), y_0 refers to the initial state probability, $P(y_{t+1}|y_t)$ is state transition probabilities and $P(x_t|y_t)$ is emission probabilities.

Linear CRF is a discriminative case of HMM. In Linear CRF, an observed variable has a connection with its corresponding hidden variable, while a hidden variable also has connections with both its previous and later hidden variables. Lafferty et al. [27] formulate the conditional probability as:

$$P(Y|X) = \frac{1}{Z(X,\Theta)}\exp\Big(\sum_t\sum_k \theta_k f_k(y_t, y_{t-1}, x_t)\Big), \tag{9}$$

where $Z(X,\Theta) = \sum_y exp(\sum_t \sum_k \theta_k f_k(y_t, y_{t-1}, x_t))$

In this equation, f_k refers to feature function and θ_k is the parameter with respect to feature k. In practice, one uses not only a single feature as input but also features like suffix, prefix, PoS and other character-based features to improve the performance of the model. Therefore, there may be more than one value of k in the summation in (9).

[5]The CRF sequence model that is discussed here and Stanford NER are linear CRF rather than generalized CRF models.

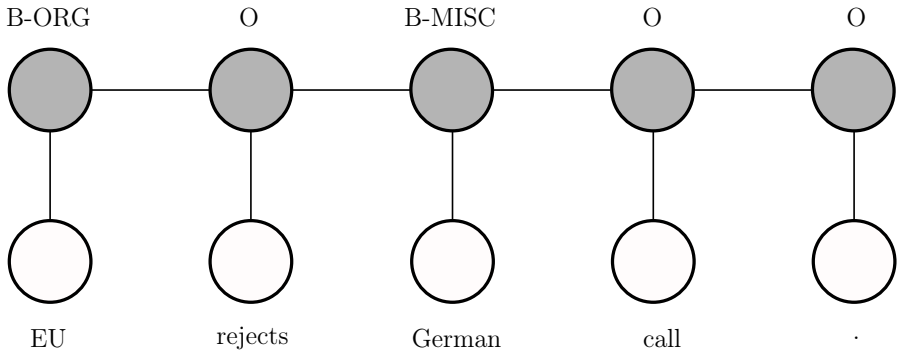

Figure 10: An example of CRF model for NER

An efficient inference algorithm for CRF is critical during the training and predicting process. For linear CRF, the algorithms are very similar to the inference algorithms for HMM: the forward-backward algorithm and the Viterbi algorithm.

The forward-backward algorithm computes the posterior marginal probabilities for all hidden variables given the observations from both forward and backward directions.

The Viterbi algorithm assigns the most likely states to hidden variables for prediction based on the result of the forward-backward algorithm. It is a dynamic programming algorithm, and its recursive equation can be obtained by utilizing part of the outcome from the forward-backward algorithm.

Detailed mathematical explanations for these two algorithms can be found in the work of Sutton et al. [40] or other textbooks. With further specifications, these algorithms can be used for linear CRF.

In practice, linear CRF can be utilized in many different fields including NLP, computer vision and bioinformatics. In NER, a sequence of tokens is the input data. The features of each token, such as PoS, and other semantic roles, can be fed into the model as well. Figure 10 shows an example of how CRF models a sequence of tokens and labels them for NER.

3.8 Bi-directional LSTM

Bi-directional LSTM (BiLSTM) is another tool for sequence modeling. Chiu and Nichols [10] proposed a BiLSTM model with multiple input features to solve the task of NER. In this subsection, I introduce this BiLSTM NER model.

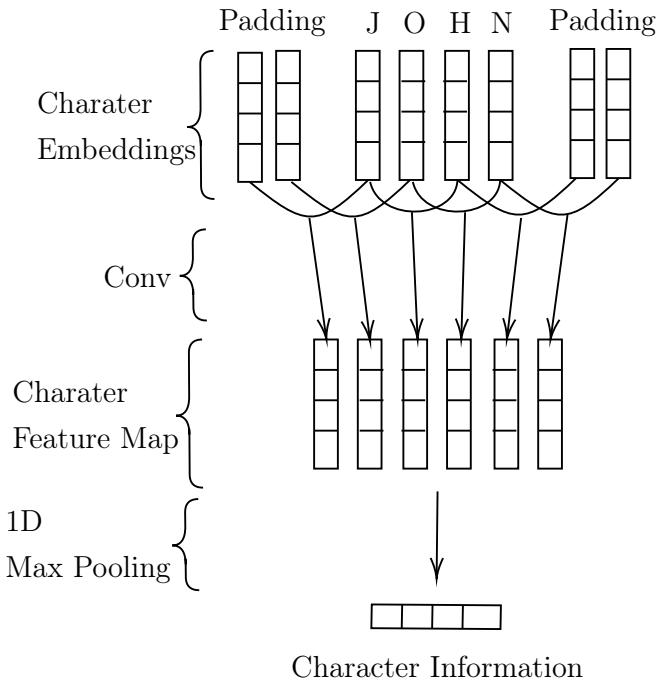

Figure 11: The extraction of character information with Conv

As introduced previously, BiLSTM can extract information from both the forward and the backward direction, which allows it to model a text sequence, since each token may have a connection with its surrounding tokens. As shown in Figure 6, in the case of NER, x_n is the embedding vector of a certain word, and h_n is a feature vector that is fed into the output layer. Eventually, each token is annotated.

Chiu and Nichols [10] feed a concatenation of four features as the input to their BiLSTM NER model: word representation, character representation, case feature and lexicon feature.

Character representation is introduced to extract the suffix and prefix information of a word. To achieve this goal, character embeddings are randomly initialized. As shown in Figure 11, they pass through the Convolutional layer and 1D Max Pooling to extract the character information.

Case feature can be as follows:

- Fully capitalized token, such as: “USA”, “NASA”
- Tokens with only the initial letter capitalized, such as: “Donald”
- Fully lower case token

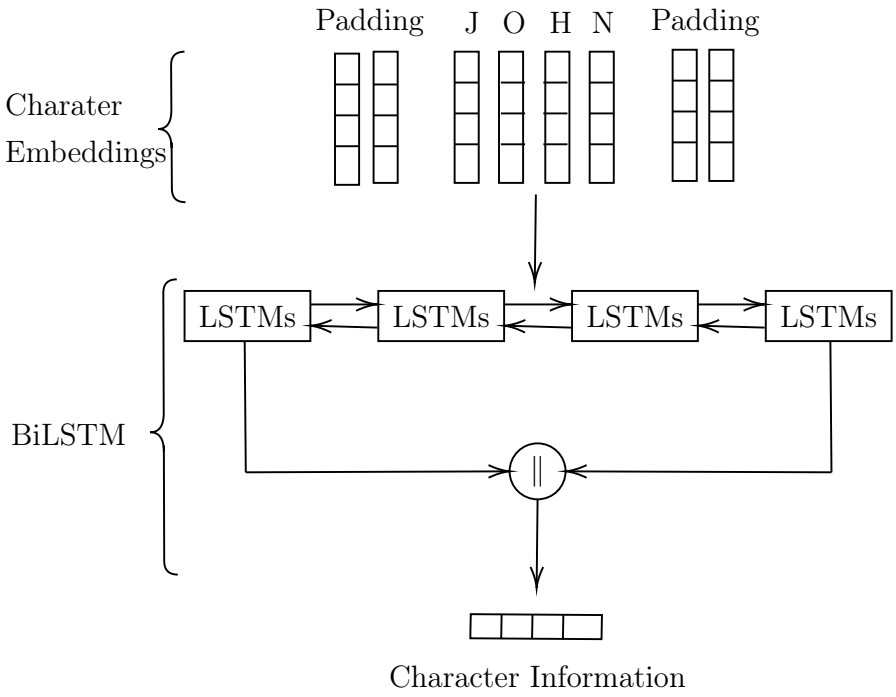

Figure 12: The extraction of character information with BiLSTM

- Tokens with mixed uppercase and lowercase, such as: “al-Qaeda”
- Punctuation token
- Other

Case feature is encoded in a similar way as embedding and transformed into a vector through a Look-up table. The table can be trained with back-propagation too [11]. After vectors are transformed, they can be fed into the Conv layer and the Max Pooling layer. The resulting feature map is used for further processing.

Lexicon feature is constructed by Chiu and Nichols [10] from a list of known named entities in DBpedia [3] on the basis of the categories defined in CoNLL-2003 NER shared task.

According to Chiu and Nichols [10], the contextualized input with additional information outperforms not only the regular feed-forward neural network but also the BiLSTM with only word embeddings.

3.9 BiLSTM-CRF

BiLSTM and CRF have been proved been effective on the task of NER. What about a hybrid model combining both models? Ma and Hovy [31] and Lample et al. [29] have proposed two models with BiLSTM layers and CRF output layer. Both of them contain character information as a feature. In the model of Ma and Hovy [31], character information is obtained in the same way as Chiu and Nichols [10], while in the model of Lample et al. [29], it is obtained by BiLSTM. Figure 12 shows the architecture of the neural network to obtain the character information.[6] Additionally, the model of Ma and Hovy [31] applies case information as a feature, while the model of Lample et al. [29] does not. All the features are concatenated and fed into BiLSTM layers. Eventually, the output is obtained by decoding the result of the CRF layer.[7]

Reimers and Gurevych [37] evaluate the performance of the models proposed by Ma and Hovy [31] and Lample et al. [29] with a different parameter setup. They claim that the model proposed by Ma and Hovy [31] can be trained within much less time than the model proposed by Lample et al. [29], while their performance is comparable. Therefore, my master's thesis is inspired by Ma and Hovy [31] and mostly follows its implementation. Figure 13 shows the general network structure.

All the word embeddings used by Lample et al. [29], Ma and Hovy [31] and Reimers and Gurevych [37] are context-independent, such as Word2Vec, which only map a token to a unique embedding vector. Peters et al. [36] include ELMo embedding, which is context-dependent, as an additional feature and feed enhanced embedding into BiLSTM-CRF. Akbik et al. [1] uses only the Flair embedding as the input for BiLSTM-CRF after concatenating the character-level embeddings for each token.

Figure 14 is a diagram modified on the basis of Akbik et al. [1] to show the structure of Flair-BiLSTM-CRF.[8] BiLSTM-CRF with context-independent embedding, such as Word2Vec, can get access to the information only within the scope of context (a sentence). With Flair, global information about the entire article is implicitly included.

[6] The symbol ‖ refers to the concatenation of input vectors in this thesis.

[7] Reimers and Gurevych [37] also include *numeric case* as one type of case information.

[8] Character language model refers to Flair here.

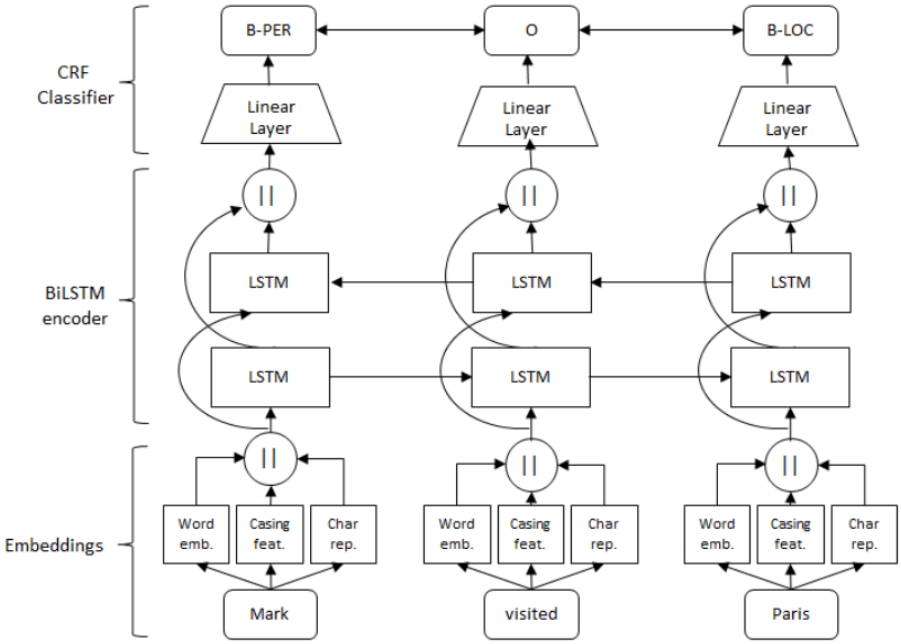

Figure 13: BiLSTM-CRF network structure [37]

3.10 NER by Pattern Mining

In addition to the probabilistic approaches and the neural network based approaches for NER, pattern mining can also be used as an approach for NER. It assigns the type of a named entity based on patterns of phrases. For example, for a sentence: "Presidential candidate x visited y last month", we can infer that named entity x is a person, and y is a location.

In PULS [16], a domain-specific pattern-based approach is used. Although the goal of the pattern mining here is to learn the event described by a sentence, it can also be utilized for NER.

Du et al. [16] first used an Apriori-like algorithm [6] to find raw patterns of high frequency. The frequency of a pattern is obtained in the following step:

1. Remove stop words and punctuation.
2. Replace names with their type. Du et al. [16] used a name dictionary and another existing NER module to obtain the data for further pattern mining.
3. Represent a sentence as a *transaction* T of $\langle W_1, W_2, W_3, ..., W_n \rangle$.
4. For a *transaction* $\langle W_1, W_2, W_3 \rangle$, only sequential patterns are allowed, namely $\langle W_1, W_2 \rangle$, $\langle W_2, W_3 \rangle$ or $\langle W_1, W_2, W_3 \rangle$. Pattern $\langle W_1, W_3 \rangle$ is not allowed.

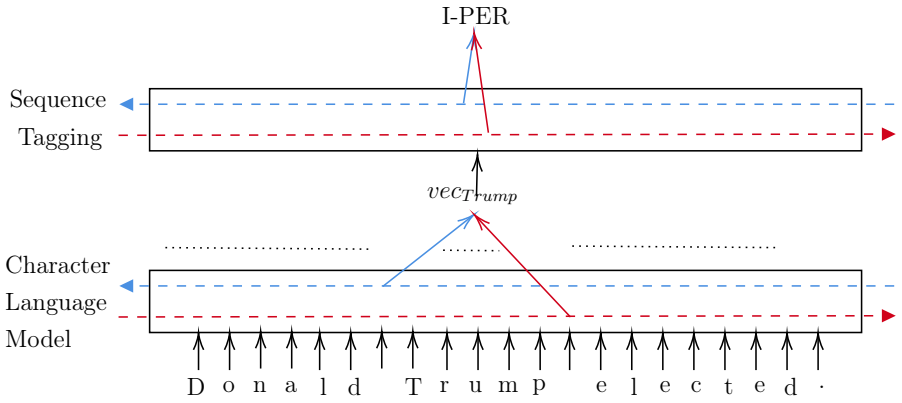

Figure 14: Flair-BiLSTM-CRF network structure

5. The candidate count $|P|$ will increase if pattern P can be found in different *transaction*.

6. The support $S_p = |P|/|T|$, where $|T|$ is the number of *transactions*.

The idea is to keep frequent patterns only ($S_p \geq S_{min}$, where S_{min} is a threshold). The frequent patterns are then manually filtered according to validity and scenario.

3.11 Pre-processing Components for NLP

Although NER approaches are essential to this thesis, pre-processing is also of great importance for my project. At the end of this section, I will introduce some key components that are used for pre-processing.

The raw input data for most NLP systems are unstructured text strings, such as articles that we can read in a book or newspaper. A pre-processing pipeline transforms the raw data to add some structure to it, so that the data can be further analyzed. There are three key components in the pipeline: **tokenizer**, **syntactic parser** and **morphological analyzer**. Their functions are listed as follows:

- A tokenizer breaks a sentence into tokens.
- A syntactic parser parses the relationship of each token based on lexicon information and assigns each token its grammatical role. It also splits sentences.
- A morphological analyzer takes the surface form of a token as input and outputs its base form, PoS, tense and other grammatical features.

For English, the tokenizer, syntactic parser and morphological analyzer are well-studied. Many are freely available, such as NLTK.[9] For my project, the integrated pre-processing pipeline in PULS project [16] is used.

For Finnish, our tokenizer and syntactic parser are the Turku Dependency Treebank [21], which contains more than 200,000 tokens from 10 text sources. Internally, the dependency treebank is based on the well-known Stanford dependency scheme [13] and a statistical dependency parser [7]. For the sentence "Sen odotetaan olevan 0,69 euroa osaketta kohti.", here is one simplified output example of the parser:

```
Sen         se        PRON   Case=Gen|Number=Sing|...   2   dobj
odotetaan   odottaa   VERB   Mood=Ind|Tense=Pres|...    0   root
olevan      olla      VERB   Case=Gen|Degree=Pos|...    2   xcomp:ds
0           0         NUM    NumType=Card               7   nummod
,           ,         PUNCT  _                          4   punct
69          69        NUM    NumType=Card               4   conj
euroa       euro      NOUN   Case=Par|Number=Sing       8   nmod
osaketta    osake     NOUN   Case=Par|Number=Sing       3   nmod
kohti       kohti     ADP    AdpType=Post               8   case
.           .         PUNCT  _                          2   punct
```

Though the direct output of this Finnish dependency parser is hard to read, it is noticeable that the parser tokenizes and extracts a syntactic tree from the sentence as indicated by the numbers of the second last column. The parser we use outputs PoS and lemma of each token as well. However, we do not use this information, because this parser may sometimes give wrong partitioning for compound words due to ambiguity. Since it gives only one resolution for each token rather than all possible candidates, it is difficult to disambiguate based on the output of the parser.

To get a full analysis for each token, a Finnish morphological analyzer is therefore applied in my thesis. This analyzer is a combination of morphological lexicon and a loss-less hyper-minimization algorithm [15, 33]. Here is one example output of our Finnish morphological analyzer:

```
[Finnish-analyser]> valtakirjassa
{'analyses': {'Finnish': [[{'base': 'valtakirja',
```

[9] https://www.nltk.org/

```
                              'pos': 'Noun',
                              'tags': {'CASE': 'Nom',
                                       'NUMBER': 'Sg'}}],
                             [{'base': 'valta',
                              'canon': 'valta+kirja',
                              'pos': 'Noun',
                              'tags': {'CASE': 'Nom',
                                       'NUMBER': 'Sg'}},
                              {'base': 'kirja',
                              'pos': 'Noun',
                              'tags': {'CASE': 'Nom',
                                       'NUMBER': 'Sg'}}],
                             [{'base': 'valta',
                              'canon': 'valta+kirja',
                              'pos': 'Noun',
                              'tags': {'UNKNOWN': ['Pref']}},
                              {'base': 'kirja',
                              'pos': 'Noun',
                              'tags': {'CASE': 'Nom',
                                       'NUMBER': 'Sg'}}]]},
 'surface': 'valtakirjassa'}
```

As demonstrated in this example, the Finnish morphological analyzer not only gives the PoS information and the lemma of the word 'valtakirjassa', but also finds all possible partitionings for this compound word. In Finnish, the last part of a lemma has the main influence on the meaning and PoS of a word. Therefore, if this word is out-of-vocabulary, we can utilize the word embedding and PoS of its last part as compensation.

4 Automatic Annotation Pipeline

In this section, I discuss my automatic annotation pipeline. The idea of this pipeline is to use English named entities, annotated by an existing English NER tagger, as the source of annotation and project the *types* of the named entities from English to Finnish. This projection is done by resolving and matching the base forms of named entities.

4.1 Raw Data Source

English news gathering English news are collected by PULS system [16] from over 3,000 sources. Over 5,000 documents are gathered daily.[10]

Finnish news gathering Finnish news are collected from Helsingin Sanomat and Yle. Around 200 documents are collected every day.[11]

4.2 Name Pre-processing

English text processing Each document collected by the system is processed by a cascade of pre-processing classifiers, including a pattern-based named entity tagger. Here, the base forms of names and their types are obtained, which are later used for projection.

The performance, especially the *precision*, of the English NER tagger is therefore crucial for the entire pipeline. It is worth pointing out that the precision of the English NER tagger controls the quality of the projected Finnish data. The recall, on the other hand, determines the variety of the projected named entities. Lower recall rate can be compensated by feeding in more news articles. Therefore, in this thesis, the precision of the English NER taggers is considered to be more essential than the recall or the overall F1 score.

For comparison, I also trained two BiLSTM-CRF models [31, 29, 37] from scratch, using Word2Vec and GloVe word embeddings. These models were trained on the CoNLL2003 English dataset. Table 2 shows an evaluation of all three English NER taggers on the CoNLL2003 test dataset

As illustrated in Table 2, BiLSTM-CRF English NER taggers that trained with CoNLL2003 have a high f-score. However, the results are different from what was reported in the papers of Ma and Hovy [31], Reimers and Gurevych [37], since I used a default hyper-parameter setup, rather than using the fine-tuned setup in the papers.

The PULS NER tagger has a worse performance if compared to BiLSTM-CRF English NER taggers. I should mention that the PULS NLP system has different tokenization compared to the CoNLL dataset, and our pattern-based NER tagger

[10] `http://newsweb.cs.helsinki.fi/`

[11] `http://hsdemo.cs.helsinki.fi/`

Source	prec	rec	F1
PULS	0.68	0.37	0.30
BiLSTM-CRF-GloVe	0.87	0.85	0.85
BiLSTM-CRF-W2V	0.89	0.90	0.89

Table 2: The quality of English NER taggers on ConLL2003 test dataset. "PULS" refers to the PULS pattern-based English NER tagger, while "BiLSTM-CRF-GloVe" and "BiLSTM-CRF-W2V" refer to the BiLSTM-CRF model trained with Glove and Word2Vec respectively.

is customized for the *business* news domain. Though the output of our pattern-based tagger is aligned to be comparable with the CoNLL dataset, the content of its test dataset, which is mostly sports news, is still skewed against our tagger. In practice, our tagger achieves higher precision on business news. To confirm this, I evaluate the PULS tagger on 10 randomly selected articles, containing both general and business news. Although this is a simple experiment, the overall precision of the PULS tagger increases to 77%.

Finnish text pre-processing The Turku dependency parser [21] has been applied for sentence splitting and tokenization. Three different problems need to be solved so that all potential names can be extracted in these steps: name identification, base form selection, and name merging.

Name identification For identifying whether a token is a name or part of a name, I use the rules based on the position of tokens as follows:

- Any capitalized token which appears in the middle of a sentence is definitely a name or part of the name.
- If token A is a name according to the previous rule, and token B, having the same base form as token A, appears at the beginning of a sentence, I assume token B is the same name.
- If a token of a potential name appears in the document only at the beginning of sentences, it is not certain and therefore not assumed to be a name.

Base form selection To determine the base form corresponding to a surface form found in a text, I consider all base forms returned by our morphological analyzer [33],

and a simple rule-based stemmer, and look through the entire article. If there is an intersection between the possible base forms of two name tokens, their true base form can then be resolved. When the intersection has only one base form, it can be confirmed to be the base form of a name directly. Otherwise, all of them will be recorded as potential base forms of the name. All potential base forms will be further filtered when matching with English named entities during projection.

Suppose the surface form "Trumpille" (in the allative case) and "Trump" (nominative) both exist in an article. Without any external knowledge, the Finnish surface form "Trumpille" will be assigned two potential lemmas by our stemmer: "Trumpi" and "Trump" (both of these lemmas have the same allative form). For the surface form "Trump", only "Trump" will be returned as a potential lemma. Then, in this case, their intersection, "Trump", will be confirmed to be the lemma of both "Trumpille" and "Trump". However, if instead the article only contains the surface forms "Trumpille" and "Trumpin" (genitive). Then both lemmas "Trumpi" and "Trump" will be recorded as potential lemmas.

We perform name identification and base form selection jointly, since they are connected, by searching for the common base forms of tokens.

Name merging I use a set of rules to merge names that consist of more than one token. Potential names can contain only the following kinds of tokens in positions other than the final position:

- Singular common noun or proper noun which must be in the nominative case, for example: "Spring Harbour".[12]
- English function words: e.g., "the", "and", "new", etc. For example: "The New York Times"
- Having no valid analyses returned by the Finnish morphological analyzer, and its surface form can be confirmed to be its own lemma during the base form selection process above.
  One example is the token "Trump" in "Trump Towerin" (genitive: "of the Trump Tower"). Our Finnish morphological analyzer will reject (not recognize) the input token "Trump". I assume that "Trump" can be confirmed as a name or as part of a multi-token name according to the rules. "Trump" can be

[12]Names such as "Helsingin Sanomat" (name of a major newspaper in Finland), where the first token is in the *genitive* case (of "Helsinki") are currently not handled by these rules, and are handled separately by a list.

confirmed to be its own base form, which means its base form happens to be the same as its surface form. In this case, "Trump" will be merged with the following token "Towerin".

We should note that when several potential name tokens are strung together, the true partitioning of names is ambiguous. During name merging, all different partitionings and potential forms of the base forms of names are cached as candidates for the following name resolution step.

Hyphenating between tokens is also a criterion for merging names, such as the Indian surname "Ankalikar-Tikekar".

4.3 Name Projection

In the next stage, I annotate Finnish names by utilizing the potential names candidates produced by the previous three steps, namely name identification, base form selection, and name merging.

The fundamental assumption is that *a name refers to only one entity in a given article* (The "One Sense Per Discourse" Assumption [43]). This assumption is expected to hold for well-edited news articles. This means that if only one instance (surface form) of a particular name has been annotated in an article, the remaining occurrences of the same name in the article—possibly involving other surface forms—can be annotated as well.

Two sets of named entities are gathered from Finnish document and English documents:

- For a Finnish document, published on day t, the previous three steps, name identification, base form selection and name merging, are used to obtain a set of potential Finnish named entity candidates, including both potential base forms and confirmed base form of names.
- From English news in the time interval ($t \pm 2$ days), using an English NER tagger, a set of English named entities and their corresponding tags are obtained. Each of them has its base form resolved by the pre-processing pipeline in PULS.

Names can naturally be matched according to their base form. The type of the English named entities can, therefore, be projected to their Finnish counterparts.

The remaining Finnish names candidates, for which no type annotation can be inferred, are dropped after this step.

The idea of a time window ($t \pm 2$ days) is to take advantage of the fact that names overlap significantly in different articles due to continuous coverage of important events, and therefore optimize the memory usage and time efficiency.

Again, take "Trump" as an example. Suppose I have a named entity "Donald Trump" from the English news articles and it is recognized as "Person". I may have "Donald Trumpille" in a Finnish article; if the surface form "Trump" is not present in the same Finnish article, as I mentioned already, I can only infer that the base form of "Trumpille" is "Trumpi" or "Trump", using stemming rules. In addition, "Donald Trumpille" has two tokens but I do not yet know whether they belong together as one name. Therefore, "Donald", "Donald Trump", "Donald Trumpi", "Trump" and "Trumpi" are all Finnish name *candidates*. After matching, only "Donald Trump" will be kept and annotated as *Person*, while other candidates, namely "Donald", "Donald Trumpi", "Trump" and "Trumpi", are dropped.

In addition, for the *Person* type only, names are connected by their partial base form. Once "Donald Trump" gets annotated, all the other "Donald" and "Trump" tokens in the entire article will be annotated as *Person* as well.

4.4 Special cases: rule-based projection

I use extra steps to handle special cases in this process. In Finnish, geo-location names, such as the names of countries, are often different from their English names. For example, France is "Ranska" in Finnish, and the United States is "Yhdysvallat" in Finnish. Some organizations also have the same problem, as UN is "YK" in Finnish, etc. Therefore, I manually build a small database of frequent names, including Finnish geo-locations, and a few of the major and most frequently occurring international companies and organizations, to assure that they are annotated correctly. In addition, this covers some cases which the English tagger fails to catch. I also filter out names that can have multiple types, such as MacLaren, since these are ambiguous.

Additionally, I introduce a list of 1000 common first names and assume that names beginning with these tokens are of type *Person*. However, this practice requires more rules to constrain its outcome:

- A Person name should have at most 2 tokens.
- A Person name should not start with "The".
- No token in a Person name should be fully uppercase.
- A Person name should be mentioned using the full name at least once in the article.

These rules are simple, naive and strict. The purpose of these rules is to remove any uncertain instances and make the data as clean as possible. Even if only one name in an article can meet all these rules, all other name instances related to that name instance will be correctly annotated. Also, taking advantage of our enormous amount of data, I can afford to filter out uncertain data without worrying about the amount of remaining data.

Currently, the annotations may be wrong when an article only mentions the last name of a person, which also happens to be the name of a location. For example, "Sipilä" is the last name of the former Prime Minister of Finland, and may, therefore, be mentioned many times in an article, without mentioning the full name, "Juha Sipilä". Coincidentally, "Sipilä" is a town in Finland. The situation where both the person and the location are mentioned in the same article rarely occurs in practice and can be tackled by filtering out such names.

5 NER Model

In this section, the details of the adapted BiLSTM-CRF model for Proj-NER and the hyperparameter setup for training this model are introduced. The basic network structure of the model is inspired by [31, 37]. The model is implemented in Keras with TensorFlow as its backend. The CRF layer is provided by Keras-contrib[13]. The training process runs on an Nvidia GeForce 1080 Ti GPU. It took around 3 hours to train the model using the setup in this section. The model is shown in Figure 15.

As seen in Figure 15, Part-of-Speech (PoS) is included as an additional feature, compared to the model of [31]. This is because a lemma may be assigned multiple PoS tags by our morphological analyzer [33]. Word embeddings such as Word2Vec [32] may implicitly contain PoS information but will still be static regardless of context.

[13] `https://github.com/keras-team/keras-contrib`

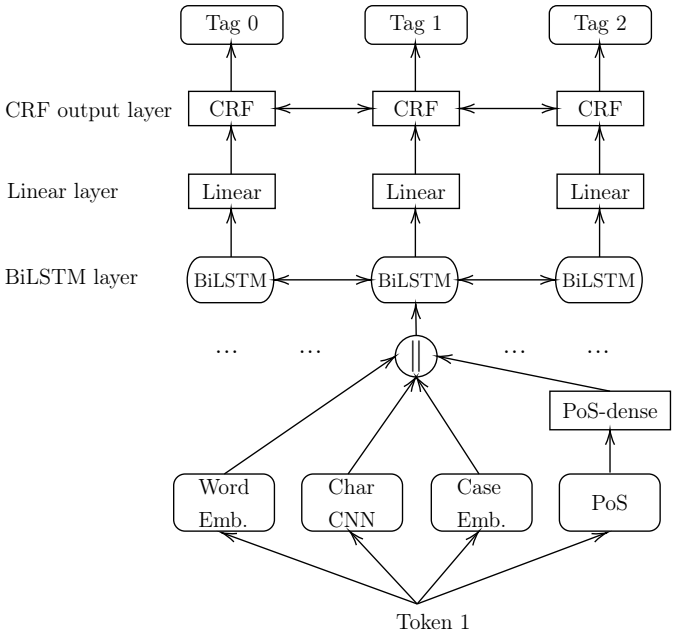

Figure 15: Adapted BiLSTM-CRF network structure for Proj-NER

Using PoS as an input feature also compensates for the out-of-vocabulary problem in embeddings. In these cases, not even the implicit PoS information can be detected by the network if PoS is not a part of input features.

5.1 Data Encoding

Tokens are encoded into to several features: *word embedding*, *character embedding*, *case embedding* and *Part of Speech* (PoS). Except for PoS, most of the features follow the setup in [37]. Word embeddings are extended with a special mark for ambiguous tokens—tokens, for which our morphological analyzer fails to return more than one base forms and PoS. These tokens are replaced with a special token "AMBIGUOUS". Additionally, I only use the embedding of the last part of a compound word if this word is out-of-vocabulary. This is because the last part is the essential part of compounds in Finnish. Character embeddings are extended with a special value for "unrecognized" character. The PoS feature is encoded as an array of ones and zeros. Each dimension corresponds to one PoS type, including PADDING and UNKNOWN. Integer "1" is assigned to the dimension corresponding to the token's PoS. If a token is a compound word, only the PoS of its last part is used for encoding. If a token has multiple PoS analyses, more than one position in the PoS array is assigned "1". The values of PoS are as follows:

- PADDING
- UNKNOWN
- Noun
- Verb
- Adj
- Adv
- Pron
- Conj
- Interj
- Num
- Punct
- other

5.2 Parameter Initialization

Word Embedding I am using a pre-trained Word2Vec embedding matrix trained by [28]. It has been trained on 4.5B words. As mentioned previously, vectors for "PADDING", "UNKNOWN" and "AMBIGUOUS" tokens are included. Both "UNKNOWN" and "AMBIGUOUS" tokens are randomly initialized with the uniform sampling from -0.25 to 0.25, while "PADDING" embedding is a zero vector.

Character embedding Character embeddings, including "UNKNOWN" character embedding, are randomly initialized with uniform samples from $-\sqrt{\frac{3}{dim}}$ to $\sqrt{\frac{3}{dim}}$, where $dim = 30$.

Case embedding Case embeddings are randomly initialized using the Keras defaults, which is applying a uniform initializer. And the dimensionality of the case embeddings is 10.

Weight Matrices and Bias Vectors Weights and bias of the network follow the default setup in Keras. Most of the weights are initialized as an uniform sample from $[-\sqrt{\frac{6}{N_i+N_o}}, \sqrt{\frac{6}{N_i+N_o}}]$, where N_i and N_o refer to the number of input and output units in weight tensor respectively. Bias is initialized with zeros.

5.3 Optimization

Optimizer I used the Adam optimizer, as recommended in [37]. The setup for the Adam optimizer also followed the Keras default setting: $lr = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$. Although [31] used gradient norm clipping of 5.0, I did not.

Early stopping and learning rate decay I applied early stopping following the categorical accuracy on the training dataset in case of over-fitting. On average, the training process stops after 5 epochs. I have also explored reducing the learning rate during the training process if the accuracy stops improving. However, this made the training slower, and did not improve the final result on the validation dataset.

5.4 Hyper-parameter Setup

Most of hyper-parameters values, shown in Table 3, follow the recommendations in [37]. The layer called "PoS-dense" in Figure 15 is a dense layer with a non-linear activation function, rather than an embedding layer, due to the encoding method of the PoS features, as explained in Section 5.1. For the mini-batch size, the authors recommend using the batch size between 8 and 32, depending on the size of the training dataset. However, that is the result on the CoNLL-2003 dataset, which is an English dataset. I use 50 similarly to the German NER model in [37].

I should mention that the CRF layer implemented by the *Keras-contrib* package offers two different modes for the training and testing processes: "Join" and "Marginal" for training, "Viterbi" and "Marginal" for testing. The "Join" training mode and "Viterbi" testing mode follows the "vanilla" fitting algorithms for linear CRF. "Marginal" training is optimized via composition likelihood (product of marginal likelihood), which is not optimal in this case. "Marginal" testing mode will decode the input sequence according to the training result and compute marginal probabilities. In this mode, it can, therefore, output a probability prediction of the classes for tokens. According to the documentation, the "Join" training mode can outperform the other training mode, and the "Viterbi" testing mode can achieve better performance than the "Marginal" testing mode, but reasonably close. In this work, I evaluate using both "Join-Marginal" and "Join-Viterbi" modes.

6 Performance Evaluation

In this section, I report the performance of the automatic projection pipeline and the NER model. F1-score is used as the evaluation metric. The overall F1-score is the weighted average F1-score of each category.

Layer	*Hyper-parameter*	*Number*
Char CNN	Number of filters	30
	Filter size	3
PoS-dense	Unit size	30
	Activation	Relu
BiLSTM	Number of layers	2
	State size	200
	Dropout rate	0.25

Table 3: Table of hyper-parameter for experiments

6.1 Automatic Named Entity Annotation

Our English data spans 6 years from 2012 to 2018. Our Finnish data, on the other hand, covers nearly 30 years dating back from 2018. Although this is a large amount of data, the English news is biased, since the articles that PULS collected are mainly related to the business domain. Starting from the middle of 2017, PULS also began to collect general news. Therefore, to reduce the domain bias of our English articles, only data from the beginning of 2017 to July of 2018 is utilized for development, model training, and validation. The amount of usable English articles from that period is around 4,486,000. Finnish articles only from the same time period as the English data are used. The total usable Finnish data consists of around 83,000 articles.

As mentioned in Section 3.3.1, the IBO2 scheme is used to tag tokens. Besides the O tag, here are three types of names which have been used during projection:

- Person: B-PER, I-PER
- Location, including states: B-LOC, I-LOC
- Organization, including companies: B-ORG, I-ORG

Although "Product" tags (B-PRO and I-PRO) and "Miscellaneous" tags (B-MISC and I-MISC) are used in PULS NER and CoNLL2003, they are not considered in this thesis. One reason of this is that the PULS English NER tagger will classify a name as MISC if it fails to resolve its type. This failure will decrease the quality of my projection. Another reason is that products are tagged as MISC in CoNLL2003 dataset and mixed with other MISC names, while PRO tags refer to products only

Tag	*Prec*	*Rec*	*F1*	*Support*
B-PER	0.97	0.99	0.98	823
I-PER	0.97	0.97	0.97	668
B-LOC	0.99	0.99	0.99	341
I-LOC	1.00	0.67	0.80	3
B-ORG	0.99	0.98	0.98	536
I-ORG	1.00	0.82	0.90	78
Avg / total	0.98	0.98	0.98	2449

Table 4: Quality of the automatically annotated Finnish data that has been projected from the PULS English NER tagger (1,000 sentences)

Tag	*Prec*	*Rec*	*F1*	*Support*
B-PER	0.99	0.97	0.98	776
I-PER	0.99	0.97	0.98	639
B-LOC	0.97	0.97	0.97	376
I-LOC	0.55	0.60	0.57	10
B-ORG	0.95	0.98	0.96	587
I-ORG	0.91	0.87	0.89	92
Avg / total	0.97	0.97	0.97	2478

Table 5: Quality of the automatically annotated Finnish data that has been projected from the BiLSTM-CRF-W2V model (1,000 sentences)

in the PULS NER tagger. This difference makes the projections of the English NER taggers incomparable. Additionally, Polyglot, one of my baseline Finnish NER taggers, supports only "person", "location" and "organization" tags. Thus, only these three types of names are considered in this thesis.

To evaluate the performance of the automatic projection, I manually checked 1,000 randomly selected sentences from March 2018 to April 2018. Since my three English NER taggers have different performances, the manual evaluation is conducted separately, as shown in Table 4, Table 5 and Table 6.

6.2 NER Model

For training the NER models, I used data from 2017-01 to 2017-12 (12 months). This period contains 50,009 Finnish documents, for which I found 920,658 matching

Tag	*Prec*	*Rec*	*F1*	*Support*
B-PER	0.96	0.99	0.97	767
I-PER	0.97	0.97	0.97	632
B-LOC	0.97	0.96	0.96	403
I-LOC	0.71	0.53	0.61	19
B-ORG	0.96	0.97	0.96	639
I-ORG	0.97	0.78	0.87	101
Avg / total	0.96	0.96	0.96	2561

Table 6: Quality of the automatically annotated Finnish data that has been projected from the BiLSTM-CRF-GloVe model (1,000 sentences)

English NER Source	*Train-test mode*	*Prec*	*Rec*	*F1*	*Support*
PULS	Join-Viterbi	0.94	0.92	0.93	28858
PULS	Join-Marginal	**0.94**	**0.93**	**0.93**	28858
BiLSTM-CRF-GloVe	Join-Viterbi	0.92	0.89	0.90	34526
BiLSTM-CRF-GloVe	Join-Marginal	0.93	0.91	0.92	34526
BiLSTM-CRF-W2V	Join-Viterbi	0.87	0.83	0.84	37219
BiLSTM-CRF-W2V	Join-Marginal	0.87	0.85	0.85	37219

Table 7: Validation scores on 2018-04 to 2018-05. "PULS" and "BiLSTM-CRF" refer to the Proj-NER models that are projected from the PULS NER tagger, and English BiLSTM-CRF NER tagger respectively. "GloVe" and "W2V" indicates the embedding which the English NER tagger uses.

English documents. I filtered out annotated sentences for which the English tagger produced NER tags *other than* Person, Organization, or Location. This data produced approximately 114,000 automatically annotated sentences after filtering. For validation, I used two months: 2018-04 to 2018-05. This period contained 11,452 Finnish documents, which had 389,072 English matching documents. This data produced 23,277 automatically annotated sentences, after filtering.

Instances are projected from 3 different English NER taggers: PULS pattern-based tagger, the BiLSTM-CRF-GloVe tagger and the BiLSTM-CRF-W2V tagger. Two train-test modes are also applied for comparison. As a result, 6 different Finnish NER model are evaluated. Table 7 shows the validation performance of my Proj-NER models.

As expected, upon visual inspection, I noticed instances with incorrect "ground

NER Source	*Train-test mode*	*Prec*	*Rec*	*F1*	*Support*
PULS	Join-Viterbi	**0.89**	0.77	**0.82**	916
PULS	Join-Marginal	0.80	**0.83**	0.81	916
BiLSTM-CRF-GloVe	Join-Viterbi	0.80	0.75	0.76	916
BiLSTM-CRF-GloVe	Join-Marginal	0.79	0.74	0.75	916
BiLSTM-CRF-W2V	Join-Viterbi	0.76	0.72	0.73	916
BiLSTM-CRF-W2V	Join-Marginal	0.78	0.79	0.78	916
FiNER-data	Join-Viterbi	0.83	0.72	0.75	916
FiNER-data	Join-Marginal	0.73	0.68	0.64	916
Polyglot		0.82	0.55	0.64	916

Table 8: Test evaluation. “FiNER-data” refers to the Finnish NER models trained with data from FiNER-data. "Polyglot" entry illustrates the performance of their model on the test dataset

truth”, since the projection is not entirely clean. Despite its good overall quality, the validation performance may still differ from actual performance.

I conducted further model testing and inspection to obtain better estimates of the true performance. I sampled another set of articles from 2018-08 to 2018-10 (3 months), which is outside my automatic projection time period. For further inspection and error analysis, in Section 6.3, I randomly sampled a total of 36 articles, evenly from the following 6 sections of the newspaper:

- “Talous” (Economics)
- “Politiikka” (Politics)
- “Ulkomaat” (Foreign news)
- “Kotimaa” (Domestic news)
- “Koti” (Home)
- “Kaupunki” (The City)

The first three of these categories are more closely related to the Business domain. Again, articles are **evaluated manually**. The result is shown in Table 8. Polyglot is used as the performance baseline. Two additional Finnish NER models are trained with full FiNER-data [39], including the validation and test dataset, for better comparison.[14]

As shown in Table 8, Polyglot and the model trained with FiNER-data have better

[14] `www.github.com/mpsilfve/finer-data`

precision than most of my Finnish Proj-NER taggers, but have a worse recall rate. Overall, most of my Finnish Proj-NER taggers achieve better performance. Only one model, projected from BiLSTM-CRF-W2V in Join-Viterbi mode, performs worse than the FiNER-data model in Join-Viterbi mode. The Finnish Proj-NER tagger that is projected from the PULS pattern-based English NER tagger in Join-Viterbi mode achieves the overall best performance. This result also suggests that the dataset produced by my automatic projection pipeline is valid for model training, data of large size and various topics, while FiNER-data only covers technology-related news.

6.3 Error Analysis

Despite good overall performance on all automatically projected datasets, the average F1 score of models with the different setups is still around 77%. More work is required to improve performance. To guide future work, I performed further visual inspection to examine the predictions in general.

One major problem is that the NER model gets data from automatic projection with **limited pattern diversity**. During the training pipeline, including automatic projection, there are two reasons that may cause this problem.

Firstly, flaws still exist in the automatic projection pipeline. One flaw that shows up often during visual inspection is that the projection currently does not support named entities without any capital letters. Named entities such as "Valkoinen talo" (the White House) cannot be fully recognized at the beginning of the pipeline (because the second token is lowercase). As a result, only the token with a capitalized letter such as "Valkoinen" will be predicted as a named entity by Finnish NER tagger.

Secondly, the English data source is biased in favor of foreign business topics. Within the time period of the training data, our PULS database contains mostly business news. As a consequence, more business-related news and their named entities in Finnish news can be tagged. General news may behave differently than business news, and may contain more patterns for the NER task. To verify this conjecture, I inspect each category with the model projected from the PULS pattern-based tagger in Join-Viterbi mode. As shown in Table 9, the model can achieve better performance on average on the topics which are related to foreign business or politics news, compared to domestic or local news.

Category	prec	rec	F1	support
Talous	0.91	0.90	0.90	123
Politiikka	0.89	0.79	0.83	191
Ulkomaat	0.92	0.75	0.83	227
Kotimaa	0.90	0.70	0.78	100
Koti	0.86	0.73	0.77	167
Kaupunki	0.83	0.71	0.74	108
Overall	0.89	0.77	0.82	916

Table 9: Table of the performance for each category.

	prec	rec	F1	support
B-PER	0.88	0.70	0.78	20
I-PER	0.83	1.00	0.91	10
B-LOC	0.85	0.90	0.88	52
I-LOC	0.00	0.00	0.00	5
B-ORG	1.00	0.32	0.48	19
I-ORG	0.00	0.00	0.00	2
avg/total	0.83	0.71	0.74	108

Table 10: Table of the performance for Kaupunki.

Another major problem is due to a **flaw in data encoding**. As mentioned previously, out-of-vocabulary compound lemmas are decomposed and assigned the embedding of the last part of the compound lemma. Many ordinary tokens may benefit from this approach, while organization named entities do not. During visual inspection, I noticed that the name of some Finnish national or local governmental departments can be a made-up word or a compound word. Such names will either be assigned the "UNKNOWN" token embedding or the embedding of the last part of the compound word, which is most likely a common noun. For example, "Verohallinto" (Tax Administration) does not have an embedding as a whole. However, "hallinto" ("government") is a common noun within the embedding vocabulary. As a result, these named entities are more likely to be tagged incorrectly. As illustrated in Table 10, the performance of organizations (B-ORG) suffers from severely low recall rates due to this problem, as well as the previously mentioned problem that Finnish domestic named entities are less likely to get projections in the pipeline.[15]

[15]Because they are unlikely to appear in English-language news.

In addition to the problems of performance, we discuss additional factors that may impact the performance of Proj-NER taggers that are projected from the PULS English NER tagger. Although the performance of our PULS pattern-based NER tagger is not the best among the English NER taggers, its projected dataset and corresponding Proj-NER taggers achieve the best performance in Finnish. Two factors may cause this result:

- Most English articles in the training dataset are related to the business domain. PULS English NER tagger is specialized for business news. As a result, the performance of the PULS English NER projection is better than the other two English NER projections, where English NER taggers are trained for general-purpose NER.
- Three categories from Table 9 are most related to the business domain: Talous (Economy), Politikka (Politics), Ulkomaat (Foreign news). Although I try to have the testing articles evenly distributed among the different categories, the absolute amount of named entities from the business domain is still higher than from the other categories. As indicated in Table 9, around 59.1% of the named entities are from the business-related articles.

It is possible that the business-biased projection and the business-biased evaluation lead to the result that the PULS English NER tagger projected to the Finnish Proj-NER tagger gives the best performance. This possible effect should be explored further, but I leave it for future work.

7 Additional Experiments

7.1 Locality Extraction and Visualization

In this section, the name linking steps and setup for locality visualization is briefly introduced.

7.1.1 Name Linking

Name linking is one essential step after NER. There are three name linking approaches: name linking within a document, name linking across documents and

name linking with a knowledge base. In this thesis, I consider only name linking within a document.

The base forms of named tokens are resolved and merged in a similar rule-based way as the approach used in base form selection and name merging described in Section 4.2. Names are then linked according to their lemma. After my Proj-NER tagger has been applied, each token has a tag or probabilities of classes (tags) attached. If all tokens of the same entity are classified into the same tag, its type is determined. Otherwise, if the tags conflict, the type of the named entity is decided by averaging over all its tokens. The idea of averaging is to get better accuracy. For organizations and locations, names can be linked directly, since they are very likely to receive the same lemma.

For persons, they are further linked for their first name and last name across the whole article. For example, suppose both "Donald Trump" and "Trump" show up as named entities in a news article separately. "Donald" is a common person first name. "Donald" and its following token "Trump" are therefore be merged into one. Now, if both "Donald Trump" and "Trump" can be predicted to be of type person after averaging, they are linked and the salience for this named entity is computed according to 1.

It is still possible that several persons that share the same family name may appear in the same news. In this case, it is more likely to have their first name appearing across the article rather than their last name. In this case, merging and linking are performed for their first name only.

As mentioned before, one problem of BiLSTM-CRF model is not being capable of seeing information beyond the context window. The benefit of linking names and averaging their probabilities is that it enables the model to break the context barrier and get access to the information in the scope of an entire article.

7.1.2 Visualization

After named entities are labeled with their types, locations are filtered out and mapped to geo-coordinates. Geo-coordinate information is freely available online. In this thesis, the GeoNames geographical database[16] is used as the source. The database not only provides coordinates of locations on different scale, as global as a country or as local as a hospital, but also gives aliases of locations. This simpli-

[16] `http://www.geonames.org/`

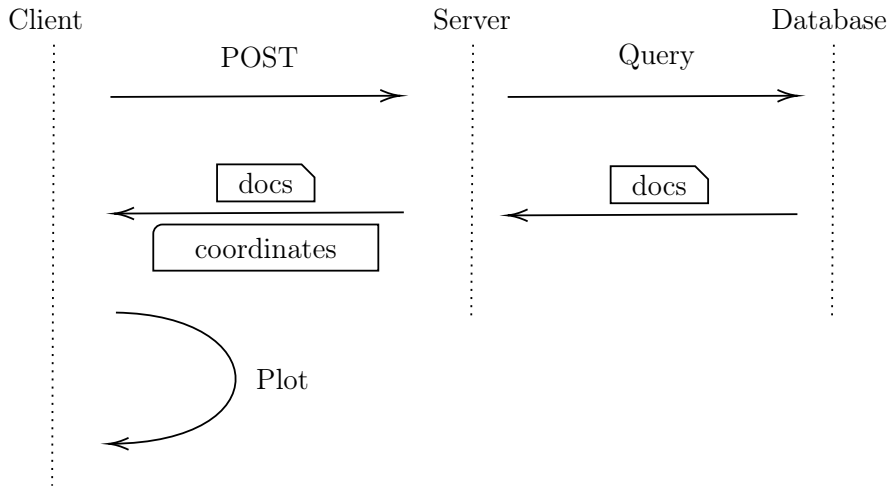

Figure 16: Network Flow of Locality Visualization

fies my work, but it also causes another problem: names can be ambiguous. For example, "Hollanti" (Holland in Finnish) can be the name of a small neighbourhood in Finland. Multiple Finnish neighbourhoods may share the same name. They can also be ambiguous with countries. Currently, ambiguous names are not used for visualization.[17] Except for ambiguous names, all Finnish locations, all countries and some capitals of major European countries are used in this application. I also map some simple aliases and acronyms to their corresponding Finnish full names in this process. For example, "USA" and "US" are mapped to Yhdysvallat (United States). To make plotting on a web page easier, all names are translated into Finnish and saved in JSON files as gazetteers, along with their corresponding coordinates.

Now that location entities are mapped to coordinates, they can be pinned in a map. This is implemented with the Plotly[18] and deployed as a new feature in Finnish PULS website [16]. To ease the visualization, rather than plugging in external map files, the Plotly embedded maps are applied here.

The discussion about the full implementation and setup for the Finnish PULS website is outside the scope of my thesis. Therefore only a brief description of the website is discussed here. The website is implemented with MongoDB, Express, AngularJS and NodeJS (MEAN).[19] When a user enters the locality interface, the client JavaScript triggers a POST request for all documents within a certain time interval, which is a year at the moment. After all documents have been collected

[17] For future work, I will try to disambiguate ambiguous locations.
[18] A data visualization framework. `https://plot.ly/`
[19] `http://meanjs.org/`

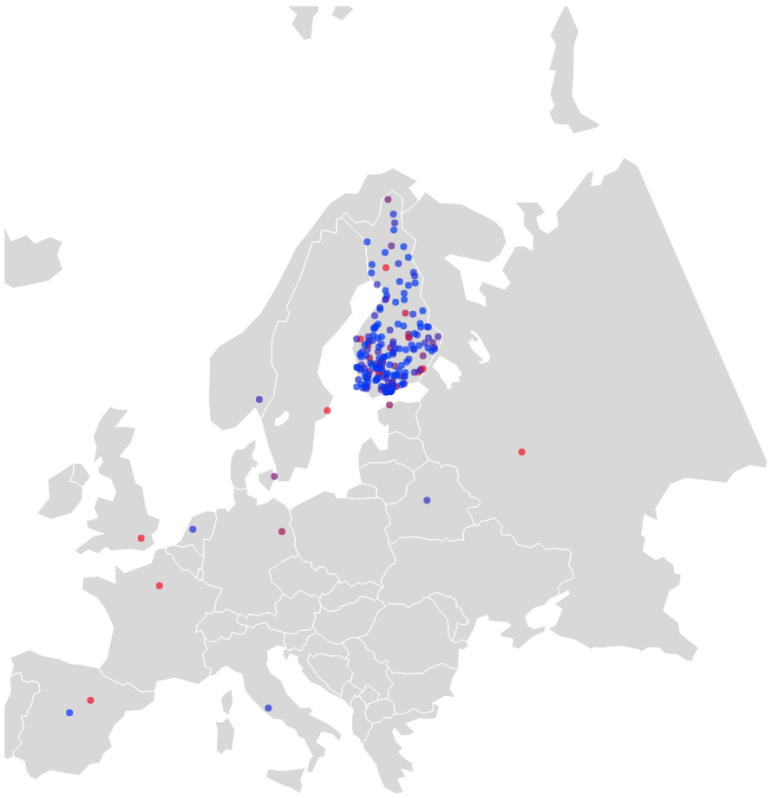

Figure 17: Heat scatter map for news locality within Europe. A red spot indicates that more news articles are related to this location, while a blue spot suggests fewer news articles.

from the back-end, the client groups them according to their locality. By utilizing the gazetteers which are mentioned above, it is easy to point news on the map according to their locality and color the points according to the amount of news. Figure 16 is a flow illustration for a single plot call from client to server.

News articles are plotted into two scales: global level and European level. As demonstrated in Figure 17 and Figure 18, each region or spot is colored according to the number of news that location is related to. Since the project is built for Finnish news, and most of the news articles are obtained from Helsingin Sanomat, it is reasonable to see that dots are much denser around the Helsinki region, and Finland is much "hotter" compared to countries like USA, Russia, and China.

If a user clicks a dot or a country in the interface, a list of news titles are displayed as Figure 19 shows.

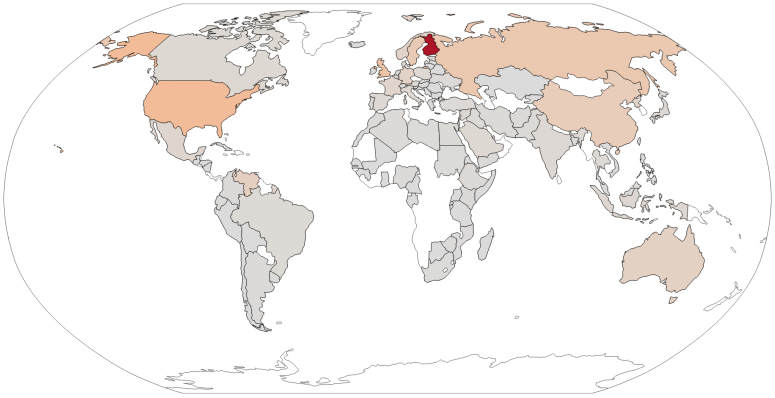

Figure 18: Heat map for News Locality worldwide. The country colored in red means that more news articles are related to it, while a grey country indicates fewer news articles.

- Maailman pienin poikavauva pääsee sairaalasta yli puolen vuoden jälkeen kotiin Japanissa
- Lehti: Suomen hävittäjähankkeessa mukana olevalla F-35:llä ollut useita hätälaskuja Japanissa
- Japanilaista lääkeprofessoria epäillään huumausaineiden valmistamisesta yhdessä oppilaidensa kanssa
- Japani aloitti ydinpolttoaineen poistamisen yhdestä Fukushiman tuhoutuneista reaktoreista
- Ensimmäiset asukkaat pääsivät palaamaan Fukushiman ydinvoimalakaupunkiin
- Suomalaisoppeja saanut Japani joutuu MM-kisojen puolivälierissä kestomenestyjän hampaisiin
- Japanin F-35-hävittäjä putosi Tyyneenmereen, lentäjä kateissa
- Japanissa joka neljäs alle nelikymppinen on neitsyt – Hyvä-palkkaiset miehet harrastavat paljon enemmän seksiä kuin pienituloiset

Figure 19: Part of news list for Japan

7.2 Named Entities in Document Representation

To evaluate the contribution of named entities to the quality of document representations, a series of experiments are conducted. In this section, the data and setup for the experiments are introduced. The outcome is also evaluated and discussed.

7.2.1 Experiments

Task This evaluation task is inspired by Chen [9]. The goal is to solve a multi-class document classification problem under these conditions:

- Ordinary NE tokens: The same NE tokens that appear in the article are used

when building the document representation.

- Unified NE token: Short names, first names and acronyms are replaced with their corresponding full names. Word embeddings of their tokens will therefore be shared.
- No NE token: All NE tokens are filtered out before building document representation.

Method The experiment follows similar procedures as those mentioned by Chen [9]. The first step is to vectorize each document. Doc2Vec is applied, since it is an out-of-the-box solution.[20] For comparison to Doc2Vec, I include document representations composed by weighting and averaging word embeddings. Four different weighting approaches are listed as follows:

- average: Naively weighting according to the number of token occurrences.
- salience: Weighting only with salience according to (1)
- tfidf: Weighting only with ordinary TF-IDF formula according to the product of (2) and (3)
- sal-tfidf: Weighting with both salience and TF-IDF according to Formula 4

Finnish Fasttext,[21] which is trained on Common Crawl[22] and Wikipedia by Facebook and provides sub-word embeddings, is used here to compensate for the out-of-vocabulary problem. Additionally, for the weighting schemes that involve TF-IDF, I compute IDF dictionaries from two different corpora for two different evaluations: the experiment corpus described in Paragraph "Data" and an external corpus that contains news from 2017.

This results in a total of 7 document representation approaches in this thesis.

After vectorization, SVM [12], implemented by scikit-learn,[23] is used as the classifier. 10% of randomly sampled data is used as training data, while the rest 90% of data is used for testing. This train-test ratio follows the same setup as Chen [9] experimented.

[20] `https://radimrehurek.com/gensim/`
[21] `https://fasttext.cc/`
[22] `http://commoncrawl.org/`
[23] `https://scikit-learn.org/`

ID	Approach
1	Doc2Vec
2	averaged Fasttext, weighted by counts
3	salience-weighted Fasttext
4	tfidf-weighted Fasttext
5	tfidf*-weighted Fasttext
6	sal-tfidf-weighted Fasttext
7	sal-tfidf*-weighted Fasttext

Table 11: Approaches to document representation. * indicates that IDF is pre-computed on the basis of the external corpus from 2017.

Data As mentioned previously, there is a huge amount of data in the PULS database. Each article is manually labeled with a category by its author. I evenly sample 4,000 articles from the following four categories:

- "Kulttuuri" (Culture)
- "Urheilu" (Sports)
- "Politiikka" (Politics)
- "Tiede" (Science)

The reason for choosing these four categories is that the content of their articles is less likely to overlap with the other three categories. Therefore, the prediction threshold is easier to obtain for machine learning algorithms.

Environment The Doc2Vec algorithm is implemented to run on CPU. The remaining document representations in Table 11 involves weighted averaging of word embeddings. Therefore, the entire experiment runs on Xeon(R) E5-1650 v4 CPU, without requiring GPUs.

7.2.2 Evaluation and Summary

As illustrated in Table 12, the classification task on the four categories is a fairly easy task with my current setup and environment. All of the document representations achieved excellent performance. The bold items, "ordinary" or "unified" NE

Document Representation	NE	prec (%)	rec (%)	F1 (%)
Doc2Vec	**ordinary**	**94.7**	**93.7**	**93.7**
	unified	93.3	93.2	93.3
	none	93.5	93.5	93.5
average	**ordinary**	**95.5**	**95.5**	**95.5**
	unified	**95.5**	**95.5**	95.4
	none	94.9	94.8	94.8
salience	**ordinary**	**95.9**	**95.8**	**95.8**
	unified	95.8	**95.8**	**95.8**
	none	94.9	94.8	94.8
tfidf	ordinary	95.7	95.7	95.7
	unified	**95.8**	**95.8**	**95.8**
	none	94.7	94.6	94.6
tfidf*	ordinary	95.6	95.5	95.5
	unified	**96.0**	**95.9**	**95.9**
	none	94.9	94.9	94.9
sal-tfidf	**ordinary**	**95.6**	**95.6**	**95.6**
	unified	95.5	95.5	95.5
	none	94.5	94.4	94.4
sal-tfidf*	ordinary	95.4	95.4	95.4
	unified	**95.8**	**95.8**	**95.8**
	none	94.8	94.8	94.8

Table 12: Performance of different document representation. * indicates that IDF is pre-computed on the basis of the external corpus from 2017.

token, outperform all of those with "none" NE token within each approach. The underlined approach—document representation with "unified" NE token and TF-IDF* weighting—achieves the best overall performance. Without any work, the baseline approaches in this experiment are Doc2Vec and "average" approach with "ordinary" NE tokens. Our best method gives error reduction of 35% compared to Doc2Vec and almost 9% compared to the "average ordinary" approach.

The result shown in Table 12 suggests that the setting of named entities weights and the representation of named entities has a positive influence on the quality of document representation in Finnish.

8 Conclusion and Future Work

Research on less-resourced languages such as Finnish often suffers from a lack of resources. One way to tackle this problem is to transfer the functionality of an existing model from a resource-rich language to a new model in a resource-poor language. In this thesis, I propose an approach for building a Finnish NER dataset by leveraging the output from an existing English NER tagger, and projecting the types of named entities from English to Finnish. The projection is done by resolving and matching the base forms of named entities. My approach can be viewed as a projection from an existing English NER tagger to a new Finnish NER tagger (Proj-NER).

In this thesis, my automatic annotation pipeline and the specification of my Finnish NER model are introduced in detail, together with all necessary background and related work. The fundamental assumption of my automatic annotation approach is that a name refers to only one entity in a given article (The "One Sense Per Discourse" Assumption [43]). Based on this assumption, the types of named entities can be projected from English to Finnish, and a Finnish NER training dataset can be generated. A BiLSTM-CRF model is then trained using this dataset.

The main contributions of this thesis are:

- My work shows that the Finnish NER dataset produced by only rule-based projection can be used for NER model training. No parallel bilingual documents are used, only projected named entities, obtained by several *monolingual* tools.
- This thesis shows the performance of my annotation pipeline and the NER model. A BiLSTM-CRF model is applied for Proj-NER, which improves on the previous state-of-the-art result in Finnish NER and sets a new benchmark.
- Although my Proj-NER tagger projected from the English BiLSTM-CRF NER taggers is not the best model for Finnish NER, the performance of most of these Proj-NER taggers is better than the baseline Finnish NER models. This suggests that the automatic annotation pipeline not only co-operates with the PULS pattern-based NER tagger but also can be generalized to work with other English NER taggers.

To show the application of my Proj-NER tagger, two additional experiments are included in this thesis:

- **Locality extraction and visualization** This experiment visualizes the locality of news articles. I resolve the locality by utilizing my Proj-NER tagger. A gazetteer of geo-locations is used to pin news articles on a map according to their locality. This experiment effectively shows news trends in both global scale and European scale.

- **Named entities in document representation** This experiment is conducted to address one question: Do named entities contribute to the quality of document representation? I evaluate the document representations *indirectly* by using them to solve a document classification problem. The result of this experiment indicates that named entities and their weighted representation do play an important role in a document representation.

For future work, I plan to first tackle the problems that I mentioned in Section 6.3. A new fully character-based embedding and transformer model may solve the out-of-vocabulary problem and yield a significant improvement in performance. Other types of bias may arise due to certain flaws in my projection pipeline. I will conduct further inspection and try to solve the bias problems next. Secondly, I plan to combine my pipeline with a disambiguation model so that both the pre-processing and the data encoding steps can be improved. Thirdly, it would be interesting to generalize my approach to other languages with limited NER tools, such as Estonian, for which corresponding news dataset can be obtained.

Acknowledgements

I would like to express my gratitude to my supervisor Prof. Roman Yangarber for his continuous support and guidance in the course of this thesis, as well as to Prof. Jyrki Kivinen for his thorough comments and suggestions. I would like to thank Dr. Mian Du and Dr. Lidia Pivovarova. Much of my work is building on their efforts. I thank the Department of Computer Science for providing a productive working environment, and especially to the technical support team, whose excellent expertise helped resolve all technical challenges. I would also like to acknowledge the contributions of my co-workers Max Koppatz and José María Hoya Quecedo, who had given their comments on different aspects of this thesis. Thanks to them and my supervisor, the results of this thesis have been published in NoDaLiDa 2019. Max and Txema, I hope you can have good progress with your own master's theses and enjoy your

time in your new jobs. In the end, my greatest gratitude goes to my family and all my friends. Thank you for your continuous trust and support.

This work was supported in part by TEKES, the Finnish Funding Agency for Technology and Innovation, Project "iMEDL—Digital Media Evolution through Deep Learning" (number 5933/31/2017), and by Helsinki Institute for Information Technology HIIT.

References

1 A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proceddings of the 27th International Conference on Computational Linguistics*, Santa Fe, USA, 2018, pp. 1638–1649.

2 R. Al-Rfou, V. Kulkarni, B. Perozzi, and S. Skiena, "Polyglot-NER: Massive multilingual named entity recognition," in *Proceedings of the 15th SIAM International Conference on Data Mining*, Vancouver, Canada, 2015, pp. 586–594.

3 S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference*, Busan, Korea, 2007, pp. 722–735.

4 A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer, and M. Auli, "Cloze-driven pretraining of self-attention networks," *CoRR*, vol. abs/1903.07785, 2019.

5 D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

6 F. Bodon, "A fast APRIORI implementation," in *Proceedings of the 3rd Workshop on Frequent Itemset Mining Implementations*, Melbourne, USA, 2003, pp. 63–74.

7 B. Bohnet, "Very high accuracy and fast dependency parsing is not a contradiction," in *Proceedings of the 23rd International Conference on Computational Linguistics*, Beijing, China, 2010, pp. 89–97.

8 P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

9 M. Chen, "Efficient vector representation for documents through corruption," *CoRR*, vol. abs/1707.02377, 2017.

10 J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 2016.

11 R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

12 C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

13 M.-C. De Marneffe and C. D. Manning, "Stanford typed dependencies manual," Stanford University, Tech. Rep., 2008.

14 J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 20th Joint Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies*, Minneapolis, USA, 2019, pp. 4171–4186.

15 S. Drobac, M. Silfverberg, and K. Lindén, "Automated lossless hyper-minimization for morphological analyzers," in *Proceedings of the 12th International Conference on Finite-State Methods and Natural Language*, Düsseldorf, Germany, 2015.

16 M. Du, L. Pivovarova, and R. Yangarber, "PULS: natural language processing for business intelligence," in *Proceedings of the 2016 Workshop on Human Language Technology*, 2016, pp. 1–8.

17 M. Ehrmann, M. Turchi, and R. Steinberger, "Building a multilingual named entity-annotated corpus using annotation projection," in *Proceedings of the 8th International Conference on Recent Advances in Natural Language Processing*, Hissar, Bulgaria, 2011, pp. 118–124.

18 J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by Gibbs sampling," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Michigan, USA, 2005, pp. 363–370.

19 A. Ghaddar and P. Langlais, "WiNER: A Wikipedia annotated corpus for named entity recognition," in *Proceedings of the 8th International Joint Conference on Natural Language Processing*, Taipei, Taiwan, 2017, pp. 413–422.

20 A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceeding of the 38th International Conference on Acoustics, Speech, and Signal Processing*, Vancouver, Canada, 2013, pp. 6645–6649.

21 K. Haverinen, J. Nyblom, T. Viljanen, V. Laippala, S. Kohonen, A. Missilä, S. Ojala, T. Salakoski, and F. Ginter, "Building the essential resources for Finnish: the Turku Dependency Treebank," *Language Resources and Evaluation*, vol. 48, no. 3, pp. 493–531, 2014.

22 J. Hou, M. Koppatz, J. M. H. Quecedo, and R. Yangarber, "Projecting named entity recognizers without annotated or parallel corpora," in *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, Turku, Finland, 2019, pp. 232–241.

23 E. Huang, R. Socher, C. Manning, and A. Ng, "Improving word representations via global context and multiple word prototypes," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju Island, Korea, 2012, pp. 873–882.

24 Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

25 J. Kazama and K. Torisawa, "Exploiting Wikipedia as external knowledge for named entity recognition," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Prague, Czech Republic, 2007, pp. 698–707.

26 S. Kim, K. Toutanova, and H. Yu, "Multilingual named entity recognition using parallel data and metadata from Wikipedia," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju Island, Korea, 2012, pp. 694–702.

27 J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the 18th International Conference on Machine Learning*, San Francisco, USA, 2001, pp. 282–289.

28 V. Laippala and F. Ginter, "Syntactic n-gram collection from a large-scale corpus of internet Finnish," in *Proceedings of the 6th International Conference on Human Language Technologies-The Baltic Perspective*, Kaunas, Lithuania, 2014, pp. 184–191.

29 G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proceedings of the 15th Joint Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies*, San Diego, USA, 2016, pp. 260–270.

30 Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China, 2014, pp. 1188–1196.

31 X. Ma and E. H. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, 2016, pp. 1064–1074.

32 T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

33 S. N. Moshagen, T. Pirinen, and T. Trosterud, "Building an open-source development infrastructure for language technology projects," in *Proceedings of the 19th Nordic Conference of Computational Linguistics*, Oslo, Norway, 2013, pp. 343–352.

34 J. Nothman, N. Ringland, W. Radford, T. Murphy, and J. R. Curran, "Learning multilingual named entity recognition from Wikipedia," *Artificial Intelligence*, vol. 194, pp. 151–175, 2013.

35 J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, 2014, pp. 1532–1543.

36 M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 17th Joint Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies*, New Orleans, USA, 2018, pp. 2227–2237.

37 N. Reimers and I. Gurevych, "Optimal hyperparameters for deep lstm-networks for sequence labeling tasks," *CoRR*, vol. abs/1707.06799, 2017.

38 A. E. Richman and P. Schone, "Mining wiki resources for multilingual named entity recognition," in *Proceedings of the 46th Annual Meeting of Association of Computational Linguistics on Human Language Technology*, Columbus, USA, 2008, pp. 1–9.

39 T. Ruokolainen, P. Kauppinen, M. Silfverberg, and K. Lindén, "A Finnish news corpus for named entity recognition," *Language Resources and Evaluation*, pp. 1–26, 2019.

40 C. Sutton, A. McCallum *et al.*, "An introduction to conditional random fields," *Foundations and Trends® in Machine Learning*, vol. 4, no. 4, pp. 267–373, 2012.

41 A. Toral and R. Munoz, "A proposal to automatically build and maintain gazetteers for named entity recognition by using Wikipedia," in *Proceedings of the Workshop on NEW TEXT Wikis and Blogs and Other Dynamic Text Sources*, Trento, Italy, 2006, pp. 56–61.

42 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, Los Angeles, USA, 2017, pp. 5998–6008.

43 D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Stroudsburg, USA, 1995, pp. 189–196.